

Classification Approaches for Microarray Gene Expression Data  
Analysis

By

Makkeyah Almoeirfi

A thesis submitted in partial fulfillment  
of the requirements for the degree of  
Master of Science (MSc) in Computational Sciences

The Faculty of Graduate Studies

Laurentian University

Sudbury, Ontario, Canada

© Makkeyah Almoeirfi, 2015

**THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE**  
**Laurentian University/Université Laurentienne**  
Faculty of Graduate Studies/Faculté des études supérieures

Title of Thesis Titre de la thèse	Classification Approaches for Microarray Gene Expression Data Analysis		
Name of Candidate Nom du candidat	Almoeirfi, Makkeyah		
Degree Diplôme	Master of Science		
Department/Program Département/Programme	Computational Sciences	Date of Defence Date de la soutenance	March 13, 2015

**APPROVED/APPROUVÉ**

Thesis Examiners/Examineurs de thèse:

Dr. Kalpdrum Passi  
(Supervisor/Directeur de thèse)

Dr. Mazen Saleh  
(Committee member/Membre du comité)

Dr. Hafida Boudjellaba  
(Committee member/Membre du comité)

Dr. Gulshan Wadhwa  
(External Examiner/Examineur externe)

Approved for the Faculty of Graduate Studies  
Approuvé pour la Faculté des études supérieures  
Dr. David Lesbarrères  
M. David Lesbarrères  
Acting Dean, Faculty of Graduate Studies  
Doyen intérimaire, Faculté des études supérieures

**ACCESSIBILITY CLAUSE AND PERMISSION TO USE**

I, **Makkeyah Almoeirfi**, hereby grant to Laurentian University and/or its agents the non-exclusive license to archive and make accessible my thesis, dissertation, or project report in whole or in part in all forms of media, now or for the duration of my copyright ownership. I retain all other ownership rights to the copyright of the thesis, dissertation or project report. I also reserve the right to use in future works (such as articles or books) all or part of this thesis, dissertation, or project report. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that this copy is being made available in this form by the authority of the copyright owner solely for the purpose of private study and research and may not be copied or reproduced except as permitted by the copyright laws without written authority from the copyright owner.

## **Abstract**

The technology of Microarray is among the vital technological advancements in bioinformatics. Usually, microarray data is characterized by noisiness as well as increased dimensionality. Therefore, data, that is finely tuned, is a requirement for conducting the microarray data analysis. Classification of biological samples represents the most performed analysis on microarray data. This study is focused on the determination of the confidence level used for the classification of a sample of an unknown gene based on microarray data. A support vector machine classifier (SVM) was applied, and the results compared with other classifiers including K-nearest neighbor (KNN) and neural network (NN). Four datasets of microarray data including leukemia data set, prostate dataset, colon dataset, and breast dataset were used in the research. Additionally, the study analyzed two different kernels of SVM. These were radial kernel and linear kernels. The analysis was conducted by varying percentages of dataset distribution coupled with training and test datasets in order to make sure that the best positive sets of data provided the best results. The 10-fold cross validation method (LOOCV) and the L1 L2 techniques of regularization were used to get solutions for the over-fitting issues as well as feature selection in classification. The ROC curve and a confusion matrix were applied in performance assessment. K-nearest neighbor and neural network classifiers were trained with similar sets of data and comparison of the results was done. The results showed that the SVM exceeded the performance and accuracy compared to other classifiers. For each set of data, support vector machine was the best functional method based on the linear kernel since it yielded better results than the other methods. The highest accuracy of colon data was 83% with SVM classifier, while the accuracy of NN with the same data was 77% and KNN was 72%. Leukemia data had the highest accuracy of 97% with SVM, 85% with NN, and 91% with KNN. For breast data, the highest accuracy was 73% with SVM-L2, while the accuracy was 56% with NN and 47% with KNN. Finally, the highest accuracy of prostate data was 80% with SVM-L1, while the accuracy was 75% with NN and 66% with KNN. It showed the highest accuracy as well as the area under curve compared to k-nearest neighbor and neural network in the three different tests.

## **Acknowledgements**

First and foremost, I would like to thank my advisor, Dr. Kalpdrum Passi, for his insightful guidance and inspiration in research, and for his consistent support and encouragement through all stages of my master study. I also thank him for his kindness and positive words, which have given me strong confidence to complete this work. Without his continued patience, I would not have been able to complete this project.

A special thanks to Dr. Hafida Boudjellaba, a member of my supervisory committee, for her suggestions and advice, which were very helpful in every step of this research. Also, I am exceptionally thankful that she took the time out of her busy schedule to answer my questions, and share her knowledge.

I would like to thank Dr. Mazen Saleh a member of my supervisory committee for reading my thesis and providing valuable feedback to improve the thesis.

I acknowledge King Abdullah, the king of Saudi Arabia, for giving Saudi women the full right to pursue their education abroad. Also, many thanks to the Saudi Cultural Bureau in Canada and the Saudi Ministry of Higher Education for their financial support of my project and education.

I would like to thank my family for their full support and endless love all along. Without my family's prayers and their patience throughout my years of education in Canada, in addition to continual encouragement by my friends, I would not have been able to push myself. I am really grateful for their help and motivation to succeed. Finally, a Special thanks to my friend Amna Al-ali for her spiritual guidance and constant encouragement.

# Table of Contents

Abstract .....	ii
Acknowledgements .....	iii
Table of Contents.....	iv
List of figures.....	vii
List of Tables.....	ix
List of APPENDIX.....	x
Abbreviations.....	xi
Chapter 1.....	1
Introduction.....	1
1.1 Introduction to Bioinformatics.....	1
1.2 Gene Expressions and Microarrays .....	2
1.2.1 Understanding gene expressions.....	2
1.2.2 Analyzing gene expression levels.....	2
1.2.3 Introduction to Microarrays.....	3
1.2.4 Distinct Types of Microarrays .....	4
1.2.5 Components of Microarray Technology.....	6
1.3 Automated Analysis of Microarray data.....	7
1.4 Classification Techniques.....	8
1.4.1 Support Vector Machine.....	8
1.4.2 Neural Networks .....	13
1.4.3 K-Nearest Neighbor (KNN).....	14
1.5 Objectives of the study and outline of the thesis.....	15
Chapter 2.....	17

Literature Review.....	17
Chapter 3.....	22
Materials and Methods.....	22
3.1 Methodology.....	22
3.1.1 Dataset selection .....	22
Dataset of Colon Cancer .....	22
Dataset of Leukemia .....	23
Dataset of Breast Cancer .....	24
Dataset of the Prostate cancer.....	25
3.1.2 Overview of the Procedure .....	26
3.1.3 Selection of training and testing dataset.....	29
3.1.3 Preprocessing of Data .....	31
3.1.4 Feature selection.....	31
3.1.5 Training the SVM.....	32
3.1.5 Linear Kernel SVM.....	33
3.1.6 Evaluated methods.....	34
3.1.7 L1, L2-Regularization.....	36
3.1.8 SVM with L1, L2 Regularization.....	37
3.1.9 Neural Networks .....	38
3.1.10 K-Nearest Neighbor (KNN) .....	40
3.1.11 Obtaining and analyzing the results.....	41
Chapter 4.....	42
Result and discussion .....	42
4.1 Colon cancer dataset:.....	42
4.1.1 Training the SVM with the colon cancer dataset .....	42
4.1.2 SVM with L1, L2- Regularization for the colon cancer dataset.....	45

4.1.3 Training Neural Network for colon cancer dataset .....	48
4.1.4 Training K-Nearest Neighbor for colon cancer dataset .....	49
4.2Leukemia cancer dataset: .....	51
4.2.1 Training the SVM with Leukemia cancer dataset.....	51
4.2.2 Training Neural Network for leukemia cancer dataset .....	54
4.2.3 Training K-Nearest Neighbor for leukemia cancer dataset.....	55
4.3Breast cancer dataset: .....	57
4.3.1 Training the SVM with Breast cancer dataset.....	57
4.3.2 SVM with L1, L2- Regularization for breast cancer dataset .....	61
4.3.3 Training Neural Network for breast cancer dataset .....	64
4.3.4 Training K-Nearest Neighbor for breast cancer dataset .....	65
4.4Prostate cancer dataset: .....	67
4.4.1 Training the SVM with Prostate cancer dataset.....	67
4.4.2 SVM with L1, L2- Regularization for prostate cancer dataset.....	70
4.4.3 Training Neural Network for prostate cancer dataset .....	73
4.3.4 Training K-Nearest Neighbor for prostate cancer dataset .....	74
4.2 Discussion.....	76
4.2.1 Evaluating the classifiers Performance .....	76
Chapter 5.....	81
CONCLUSIONS.....	81
future work .....	82
References .....	83
APPENDIX .....	88

## List of figures

Figure	Page
1.1 Microarray Chip .....	6
1.2 depicting the Microarray experiment protocol .....	8
1.3 Decision boundary and margin of SVM classifier .....	12
1.4 NN Multi-layer Perceptron (MLP) .....	17
1.5 Working of KNN classifier .....	18
3.2 Protocol used for the current study .....	33
3.3 Selection of training and testing datasets .....	35
3.4 Block diagram of NN Procedure .....	47
4.1.a Linear SVM: colon dataset (60:40) .....	51
4.1.b Linear SVM: colon dataset (70:30).....	52
4.1.c Linear SVM: colon dataset (80:20).....	52
4.1.d Linear SVM: colon dataset (90:10) .....	53
4.2.a Linear SVM with L1 regularization: colon dataset .....	54
4.2.b Linear SVM with L2 regularization: colon dataset .....	55
4.3 Neural Network: colon dataset .....	56
4.4 K-Nearest Neighbor: colon dataset.....	56
4.5.a Linear SVM: Leukemia dataset (60:40) .....	60
4.5.b Linear SVM: Leukemia dataset (70:30) .....	60
4.5.c Linear SVM: Leukemia dataset (80:20) .....	61
4.5.d Linear SVM: Leukemia dataset (90:10) .....	61
4.6 Neural Network: Leukemia dataset .....	63



4.7	K-Nearest Neighbor: Leukemia dataset .....	65
4.8.a	Linear SVM: breast dataset (60:40) .....	67
4.8.b	Linear SVM: breast dataset (70:30) .....	67
4.8.c	Linear SVM: breast dataset (80:20) .....	68
4.8.d	Linear SVM: breast dataset (90:10) .....	68
4.9.a	Linear SVM with L1 regularization: breast dataset .....	70
4.9.b	Linear SVM with L2 regularization: breast dataset .....	71
4.10	Neural Network: breast dataset .....	72
4.11	K-Nearest Neighbor: breast dataset .....	74
4.12.a	Linear SVM: prostate dataset (60:40) .....	76
4.12.b	Linear SVM: prostate dataset (70:30) .....	76
4.12.c	Linear SVM: prostate dataset (80:20) .....	77
4.12.d	Linear SVM: prostate dataset (90:10) .....	77
4.13.a	Linear SVM with L1 regularization: prostate dataset .....	79
4.13.b	Linear SVM with L2 regularization: prostate dataset .....	80
4.14	Neural Network: prostate dataset .....	81
4.15	K-Nearest Neighbor: prostate dataset .....	83
3.3.a	Block diagram of Selection training and testing.....	27

## List of Tables

Table	page
3.1 Format of Colon cancer dataset .....	29
3.2 Format of Leukemia cancer dataset. ....	30
3.3 Format of Breast cancer dataset .....	31
3.4 Format of Prostate cancer dataset .....	32
3.5 Sample's distribution of training and test data for 4 datasets .....	36
3.6 The confusion matrix for a two-class classifier .....	42
4.1 Comparison of classification accuracies for colon dataset .....	84
4.2 Comparison of classification accuracies for leukemia dataset .....	85
4.3 Comparison of classification accuracies for breast dataset .....	86
4.4 Comparison of classification accuracies for prostate dataset .....	87
4.5 Comparison of classification accuracies for all dataset .....	88

## List of APPENDIX

Appendix	page
APPENDIX A.....	88
APPENDIX B.....	100
APPENDIX C.....	102
APPENDIX D.....	104
APPENDIX E.....	105
APPENDIX F.....	106
APPENDIX G.....	108
Appendix H.....	112

## Abbreviations

SVM	Support vector machine
SVM-L1	L1 regularization with support vector machine
SVM-L2	L2 regularization with support vector machine
LOOCV	Leave One Out Cross Validation
ROC	Receiver operating characteristic
NCBI	National Center for Biotechnical Information
DNA	Deoxyribonucleic acid
BLAST	Basic Local Alignment Search Tool
RNA	Ribonucleic acid
mRNA	Messenger RNA
Affymetrix	American company that manufactures DNA microarray
cDNA	Complementary DNA
MLP	Multi-layer Perceptron
NN	Neural network
KNN	K-Nearest Neighbor
FP	False positive
FN	False negative
TP	True positive

TN	True negative
IG	Information Gain
RBF	Radial-based function
ALL	Acute lymphoblastic leukemia
AML	Acute myeloid leukemia
AUC	Area under curve
DM	breast cancer patients who within 5 years had grown distant metastases
NODM year	breast cancer patients who were free from the ailments after at least a period of 5 year

# **Chapter 1**

## **Introduction**

### **1.1 Introduction to Bioinformatics**

Bioinformatics is an evolving field that has risen from its incorporation in biology, mathematics and computer science. It mainly involves inception, management and examination of biological data mainly obtained from a substantive number of experimental test runs that may at times provide large data set. To this effect, there is need to establish comprehensive mechanisms to aid in the interpretation and processing this data and consequently producing accurate information that is needed for research and study purposes [1]. This led to the inception of bioinformatics, a discipline that integrates both biology and computer science.

Changes in the field of bioinformatics have encouraged numerous analysts in investigating the information and comprehension of the structural, comparative and practical properties. A significant percentage of the improvements have been investigation of genomes and proteins, recognizing metabolic and flagging pathways which characterize the genetic connections, advancement of microarray chip and leading microarray analyses to gauge the genetic articulation levels. The accessibility of the information on open sites and vaults made it simpler to do the research on such databases for instance, the National Center for Biotechnical Information (NCBI), is freely accessible for the scientists. NCBI gives organic information including DNA and protein arrangements and sways analysts to help their groupings to the database. Likewise, to process the information, finely tuned calculations were created throughout the years and have been made openly accessible. Some of them are BLAST and CLUSTALW calculations that perform grouping examination. Calculations to perform phylogenetic examination were additionally made accessible in general sites [2].

The development of the microarray technology can be depicted as the most fundamental innovation in the field of bioinformatics. This kind of technology is significant in the determination of values with respect to more than a thousand genes in a simultaneous manner. The expression values can be subjected to a number of experimental test runs to ascertain the

imperativeness of the tissues from which the genes were removed. One of the global acclimated methodologies of such classification is using the expression values of the genes that are derived from the microarray experiment. The study in this thesis are more focused on ascertaining the confidence levels in making classifications on the unknown gene sample based on microarray data by using the Support Vector Machines (SVM). There was the need to come up with a brief explanation on gene expression and microarray technology. This will help greatly in providing for an in depth comprehension of the problems that exist in the current classification mechanisms. Successively, a brief introduction is given on support vector machines, K-nearest neighbor and neural network classifiers.

## **1.2 Gene Expressions and Microarrays**

### **1.2.1 Understanding gene expressions**

"Gene expression is the term utilized for portraying the interpretation of data contained inside the DNA, the storehouse of hereditary data, into messenger RNA (mRNA) atoms that are then interpreted into the proteins that perform a large portion of the discriminating capacity of cells" [3]. Gene expression is a complex process that permits cells to respond to the changing inward prerequisites and furthermore to outside ecological conditions. This system controls which genes to express in a cell and furthermore to build or reduce the level of expression of genes.

### **1.2.2 Analyzing gene expression levels**

The expression levels of the genes can be directly correlated to the characteristics and behavior of the species. Research in the field of bioinformatics proposes that the reasons for a few variations from the norm are due to the change in the level of representation of qualities from the typical level. With the assistance of new age advances, we are presently ready to study the interpretation levels of a huge number of genes immediately. Along these lines, we can attempt to compare the articulation levels in normal and abnormal states. The outflow values in influenced qualities can help us measure them with normal representation qualities and thereby provide an explanation about the variation from the norm. The quantitative data of quality interpretation profiles can help support the fields of medication improvement, judgment of ailments and further understanding the working of living cells. A quality is viewed as

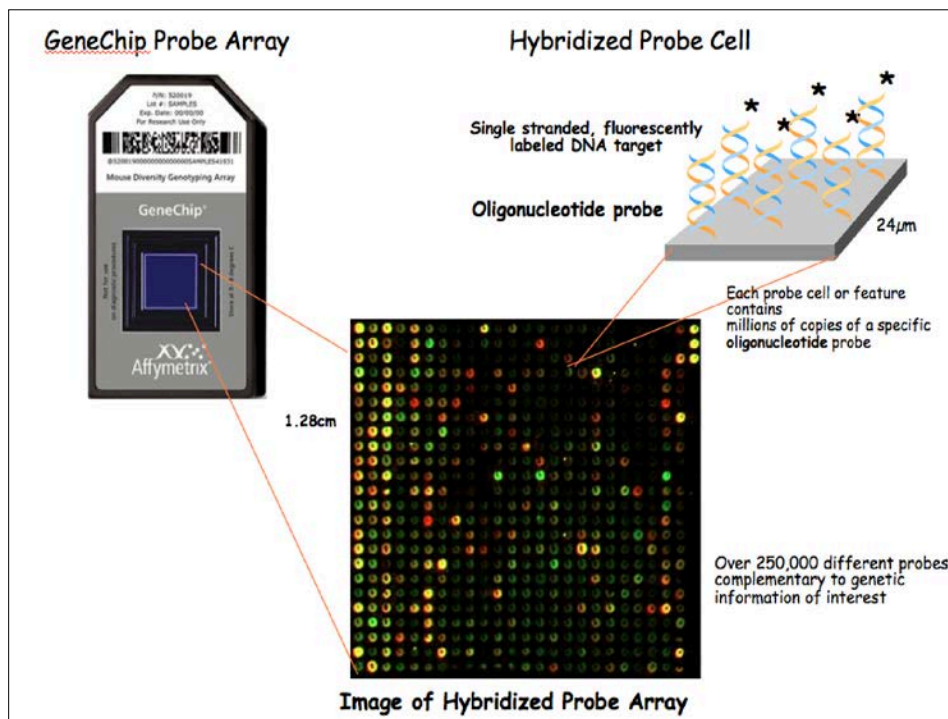
educational when its statement serves to arrange examples to an ailment condition. These educational qualities help us create grouping frameworks, which can recognize typical cells from the unusual ones. Microarray is one such instrument that can be utilized to screen the statement level of qualities [17].

### **1.2.3 Introduction to Microarrays**

A microarray is a device used to study and record the gene expressions of a large number of genes at the same time. A microarray comprises of distinctive nucleic acid probes that are artificially appended to a substrate, which can be a microchip, a glass slide or a microsphere-sized globule [4]. There are distinctive sorts of microarrays, for example, DNA microarrays, protein microarrays, tissue microarrays and carb microarrays [5]. The microarray innovation was advanced out of the need to focus measures of specific substances inside a mixture of different substances. The process was first carried out using assays. Assays were used in a variety of applications like identification of blood proteins, drug screening and so on. Immunoassays helped to focus the measure of antibodies bound to antigens in immunologic reaction forms. Fluorescent marking and radioactive naming were utilized to name either the antibodies or the antigens to which the antibodies were bound. The idea of immunoassays was later reached out to include DNA investigation. The most punctual microarrays included tests in which the specimens were spotted physically on test surfaces. The smallest attained spot sizes were 300  $\mu$  m. However, it was only when the spot sizes became smaller for accommodating more genes, that robotic and imaging equipment were deployed. Labeling methods involved using radioactive labels for known sequences. Another technique involved using fluorescent dyes to label biological molecules in sample solutions. The Southern Block technique developed later used arrays of genetic material for the first time. The mechanism encompassed labeling the DNA and RNA strand to ascertain the rest of the nucleic acid that was attached to the solid surfaces. In this procedure, denatured strands of DNA were exchanged with nitrocellulose channels for recognition by hybridization of radioactively named tests. Such an exchange was conceivable in light of the fact that denatured DNA strands structured covalent bonds with robust surfaces without re-associating with one another. These however, effectively shaped bonds with corresponding sections of RNA [17]. Southern Block strategy utilized permeable surfaces as the robust backing for DNA strands. These were later supplanted by glass surfaces, which accelerated the substance responses since the substances did not diffuse into permeable surfaces.



In 1980 the Department of Endocrinology at the University of London utilized micro spotting methods to manufacture clusters for high affectability immunoassay studies including examination of antibodies in the field of immunodiagnostics. This procedure was later received in an extensive variety of uses including organically tying examines. The result of this strategy, known as multianalyte microspot immunoassays, measured radiometric power by taking the degree of the fluorescent signs to the absorbance. The innovation has been advancing after and a ton of exploration has been fulfilled to refine this system. The main DNA Microarray chip was built at Stanford University, though Affymetrix Inc. was the first to make the licensed DNA microarray wafer chip called the Gene Chip. Figure 1.1 demonstrates a run of the mill exploration of different avenues regarding an oligonucleotide chip.



**Figure 1.1 Microarray Chip. [6]**

### 1.2.4 Distinct Types of Microarrays

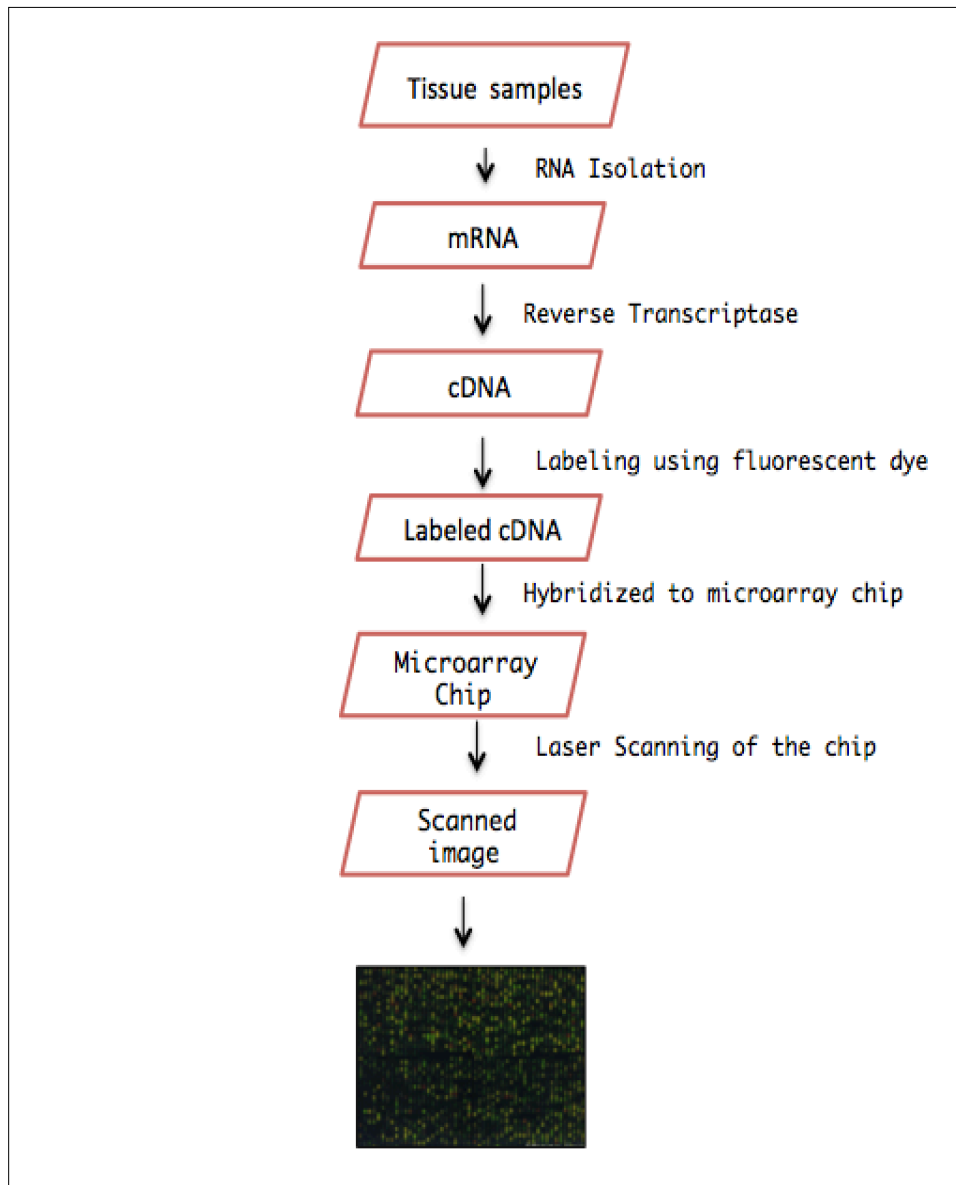
Microarrays can be divided into two types [7]:

### ***Single Channel or One Color Microarrays***

This innovation was initially presented by Affymetrix Inc. In these microarrays, individual samples are subjected through hybridization after it is named with a fluorescent color. These microarrays measure irrefutably the power of declaration. These are additionally called oligonucleotide microarrays where the tests are oligonucleotide successions that are 15 to 70 bases long. Oligonucleotides are either, incorporated independently and spotted on the chips, or they can be orchestrated specifically on the chip utilizing as a part of silicon systems. The last procedure is completed utilizing a methodology called photolithography. Tests including one color microarrays are described by effortlessness and adaptability. Hybridization of a single sample per microarray not only helps to compare between microarrays but also allows analysis between groups of samples.

### ***Dual Channel or Two color Microarrays***

These are likewise termed as cDNA Microarrays. In these microarrays, example successions and ordinary arrangements are marked with two distinctive fluorescent colors. Both these DNA arrangements are hybridized together on the DNA Microarrays and a degree of fluorescence intensities emitted by the two colors is considered so as to assess differential representation level. This outline of microarrays was created to decrease variability blunder in microarray fabrication. Hybridization of two specimens to tests on same microarray takes into account immediate correlation. These microarrays are known to be very delicate and exact. Figure 1.2 demonstrates the convention for a microarray test. Tissue examines whose quality statements are to be measured have their mRNAs (delivery person RNAs) removed. At that point reverse transcriptase is connected to change over these mRNAs to cDNAs (correlative DNAs). cDNAs are marked utilizing brilliant colors as per the specimen. At that point the example on the microarray chip is then subjected to hybridization. Amid hybridization the cDNAs tie to their reciprocal strands utilizing base pair holding. The chip is then washed to uproot unhybridized materials. Advanced picture is acquired by laser checking the chip. The picture is prepared utilizing information standardization and other picture transforming procedures to get the statement level of every quality focused around the fluctuating intensities of fluorescence.



**Figure 1.2 depicting the Microarray experiment protocol [8]**

### **1.2.5 Components of Microarray Technology**

A Microarray technology encompasses the following:

#### ***The Array:***

This is the robust base on which the hereditary material of known arrangements are organized efficiently along lattices. The process of arrangement is called spotting. The array is made up of

glass or nylon which bears a large number of wells to hold the distinctive Complementary DNA (cDNA) sequences. Each one spot on the microarray represents an independent experimental assay used to measure the presence and abundance of specific sequences in the sample strands of polynucleotides. The Arrays are comprised of glass, nylon and sometimes coated using silicon hydride. The covering empowers the microarrays to repulse water and supports hybridization of cDNA strands to the surface of the exhibit. It likewise keeps the polynucleotide tests from spreading, therefore holding commotion under wraps.

### ***Probes:***

The single stranded cDNAs that are spotted on the exhibits are known as “probes”. The target polynucleotide groupings in the organic specimen arrangements are hybridized with the complimentary tests. Adherence of probe to the array is very crucial to maintain spot integrity and prevention of the probe from being washed away during array processing. It is additionally critical as a spasmodically followed test can cause commotion to leak in subsequently decreasing the nature of resultant picture. After the test is spotted onto the show, it is air dried and presented to ultraviolet radiation to guarantee stability and solid adherence.

### ***The Spotter:***

These are mechanical instruments that apply the tests to the shows utilizing high exactness. The spotters apply each one spot to a framework on the exhibit. This aids in leading an expansive number of trials in parallel. The spotting is carried out utilizing either contact or non-contact strategies. Contact spotters have the spotting spout like an ink pen where connected weight discharges the tests on the shows. Non-contact spotters utilization ink-plane engineering or the piezoelectric narrow impacts for spotting purposes. Non-contact spotters are speedier than contact spotters; however contact spotters have more exactness as contrasted with non-reaching ones [10].

## **1.3 Automated Analysis of Microarray data**

Microarrays have made ready for analysts to assemble a great deal of data from a huge number of qualities in the meantime. The principle assignment is the investigation of this data. Looking at the size of the data retrieved from the genetic databases, we can definitely say that there is no

way to analyze and classify this information manually. In this thesis, an effort has been made to classify gene expression data of four different cancer datasets into two classes of different samples. This study tries to unveil the potential of classification by automatic machine learning methods.

## **1.4 Classification Techniques**

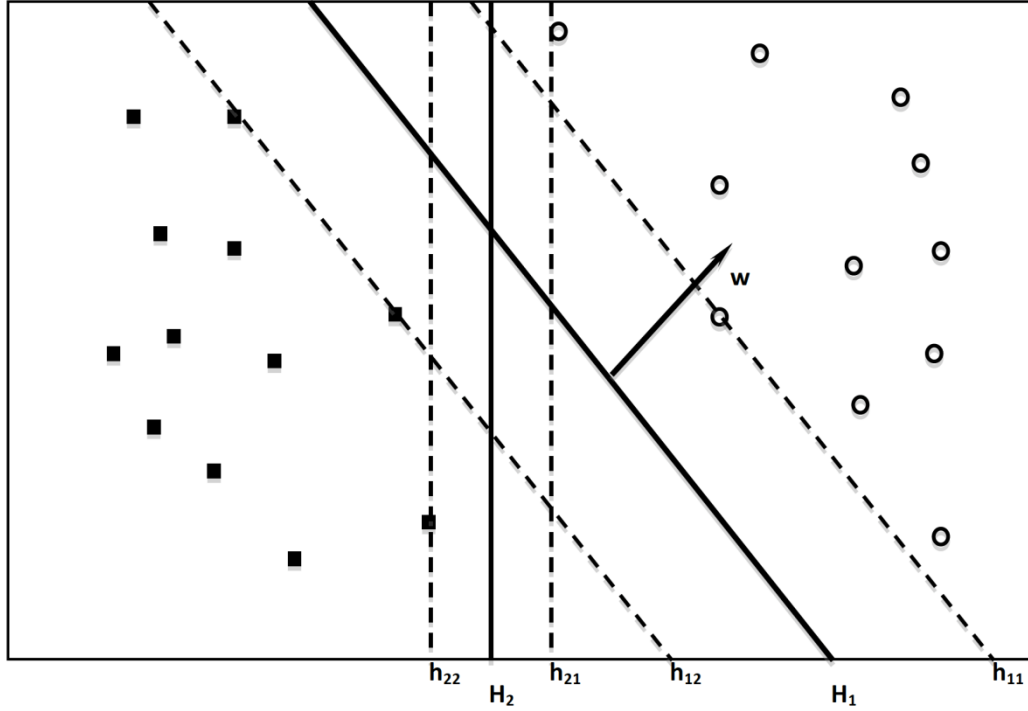
In the current study, we deal with a classification problem which focuses on dividing the samples of four microarray datasets into two categories. Any classification method uses a set of parameters to characterize each object. These features are relevant to the data being studied. Here we are talking about methods of supervised learning where we know the classes into which the items are to be ordered. We likewise have a set of items with known classes. A training set is utilized by the order projects to figure out how to arrange the items into wanted classes. This training set is utilized to choose how the parameters ought to be weighted or consolidated with one another so we can separate different classes of articles. In the application stage, the trained classifiers can be utilized to focus the classifications of articles utilizing new examples called the testing set. The different well-known to order techniques are discussed as follows [11].

### **1.4.1 Support Vector Machine**

Support vector machine (SVM) is picking up prevalence for its capacity to arrange boisterous and high dimensional information. SVM is a measurable learning calculation that groups the examples utilizing a subset of preparing specimens called support vectors. The thought behind SVM classifier is that it makes a peculiarity space utilizing the qualities as a part of the training data. It then tries to distinguish a decision boundary or a hyper-plane that differentiates the gimmick space into two parts where every half contains just the preparation information focuses fitting in with a classification. This is demonstrated in Figure 1.3

In Figure 1.3 the data points belong to one class and square points belong to another class. SVM tries to discover a hyper-plane ( $H_1$  or  $H_2$ ) that differentiates the two classifications. As demonstrated in figure there may be numerous hyper-planes that can separate the information. Taking into account "maximum margin hyper-plane" idea SVM picks the best decision boundary that differentiates the information. Every hyper-plane ( $H_i$ ) is connected with a couple of supporting hyper-planes ( $h_{i1}$  and  $h_{i2}$ ) that are parallel to the decision boundary ( $H_i$ ) and pass through the closest information point. The separation between these supporting planes is called

as margin. In the figure, despite the fact that both the hyper-planes ( $H_1$  and  $H_2$ ) isolate the information focuses,  $H_1$  has a greater margin and has a tendency to perform better for the characterization of obscure specimens than  $H_2$ . Subsequently, greater the edge is, the less the speculation blunder for the arrangement of obscure specimens is. Consequently,  $H_1$  is favored over  $H_2$ .



**Figure 1.3: Decision boundary and margin of SVM classifier [46]**

There are two sorts of SVMs, (1) Linear SVM, which differentiates the information focuses utilizing a linear decision boundary and (2) Non-linear SVM, which divides the information focuses utilizing a non-linear decision boundary. For a linear SVM the mathematical statement for the decision boundary is:

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \quad (1.1)$$

where,  $w$  and  $x$  are vectors,  $b$  is a scalar and the bearing of  $w$  is perpendicular to the linear decision boundary. Vector  $w$  is dead set utilizing the preparation dataset. For any set of information focuses ( $\mathbf{x}_i$ ) that lie above the decision boundary the mathematical statement is:

$$\mathbf{w} \cdot \mathbf{x}_i + b = k, \quad \text{where } k > 0, \quad (1.2)$$

and for the data points ( $\mathbf{x}_j$ ) which lie below the decision boundary the equation is

$$\mathbf{w} \cdot \mathbf{x}_j + b = k', \quad \text{where } k' < 0. \quad (1.3)$$

By rescaling the values of  $\mathbf{w}$  and  $b$  the equations of the two supporting hyper planes ( $h_{11}$  and  $h_{12}$ ) can be defined as

$$h_{11}: \mathbf{w} \cdot \mathbf{x} + b = 1 \quad (1.4)$$

$$h_{12}: \mathbf{w} \cdot \mathbf{x} + b = -1 \quad (1.5)$$

The distance between the two hyper planes (margin “d”) is obtained by

$$\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 2 \quad (1.6)$$

$$d = 2/\|\mathbf{w}\| \quad (1.7)$$

The target of SVM classifier is to augment the estimation of  $d$ . This target is equal to minimizing the estimation of  $\|\mathbf{w}\|^2/2$ . The estimations of  $\mathbf{w}$  and  $b$  are acquired by tackling this quadratic improvement issue under the requirements

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 1 \text{ if } y_i = 1 \quad (1.8)$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \text{ if } y_i = -1 \quad (1.9)$$

where  $y_i$  is the class variable for  $\mathbf{x}_i$ . Forcing these limitations will make SVM to place the preparation occurrences with  $y_i = 1$  above the hyper plane  $h_{11}$  and the training occasions with  $y_i = -1$  underneath the hyper plane  $h_{12}$ . The optimization problem can be explained utilizing Lagrange multiplier method. The target capacity to be minimized in the Lagrangian structure can be composed as:

$$L_P = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \alpha_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \quad (1.10)$$

$\alpha_i$  are Lagrange multipliers and  $N$  are the quantity of specimens [6]. The Lagrange multipliers ought to be non-negative ( $\alpha_i \geq 0$ ). So as to minimize the Lagrangian structure, its halfway subordinates are acquired concerning  $\mathbf{w}$  and  $b$  and are likened to zero.

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i \quad (1.11)$$

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \sum_{i=1}^N \alpha_i y_i = 0 \quad (1.12)$$

The mathematical statement is changed to its double structure by substituting the qualities from Equation 1.11 and 1.12 in the Lagrangian structure Equation 1.10. The double structure is given by:

$$L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j \quad (1.13)$$

The preparation occurrences for which the estimation of  $\alpha_i > 0$  lie on the hyper plane  $h_{11}$  or  $h_{12}$  are called support vectors. Just these training cases are utilized to get the decision boundary parameters  $\mathbf{w}$  and  $b$ . Henceforth the order of obscure examples is focused around the support vectors. At times it is desirable over misclassify some of training specimens (training error) to get decision boundary plane with most extreme edge. A decision boundary with no training mistakes however smaller margin may prompt over-fitting and can't group obscure examples accurately. Then again, a decision boundary with few training error and a bigger margin can group the obscure specimens all the more precisely. Subsequently there must be a tradeoff between the margin and the quantity of training error. The decision boundary along these lines acquired is called as soft margin. The demands for the optimization problem still hold great, however require the expansion of slack variables ( $\xi$ ), which represent the soft margin. These slack variables relate to the mistake in decision boundary. Additionally a punishment for the training error ought to be presented in the target work so as to adjust the margin worth and the quantity of training error. The target capacity for the optimization problem will be minimization of:

$$\|\mathbf{w}\|^2 / 2 + C (\sum \xi_i)^k \quad (1.14)$$

Where  $C$  and  $k$  are specified by the user and can be varied depending on the dataset. The constraints for the optimization problem will be

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 1 - \xi_i, \quad \text{if } y_i = 1, \quad (1.15)$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 + \xi_i, \quad \text{if } y_i = -1. \quad (1.16)$$



The Lagrange multiplier for soft margin varies from the Lagrange multipliers of linear decision boundary.  $\alpha_i$  values ought to be non-negative furthermore ought to be less than or equivalent to C. Henceforth the parameter C act as the upper limit for error in the decision boundary [6]. Linear SVM performs well on datasets that can be effortlessly divided by a hyper-plane into two sections. Anyhow now and then datasets are perplexing and are hard to order utilizing a linear kernel. Non-linear SVM classifiers can be utilized for such mind boggling datasets. The idea driving non-linear SVM classifier is to change the dataset into a high dimensional space where the information can be divided utilizing a linear decision boundary. In the original feature space the decision boundary is not linear. The fundamental problem with changing the dataset to higher measurement is the increment in intricacy of the classifier. Additionally the precise mapping capacity that can separate data linearly in higher dimensional space is not known. Keeping in mind the end goal to conquer this, an idea called kernel trick is utilized to change the information to higher dimensional space. In the event that  $\Phi$  is the mapping function, so as to discover the linear decision boundary in the changed higher dimensional space, attribute  $\mathbf{x}$  in the Equation 1.13 is supplanted with  $\Phi(\mathbf{x})$ . The changed Lagrangian double structure is given by:

$$L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \Phi(\mathbf{x}_j) \quad (1.17)$$

The dot product is a measure of similarity between two vectors. The key thought behind kernel trick is that it considers the dot product comparable to in the first and the changed space. Consider two data case vectors  $\mathbf{x}_i$  and  $\mathbf{x}_j$  the first space. At the point when changed to a higher measurement, they are changed to  $\Phi(\mathbf{x}_i)$  and  $\Phi(\mathbf{x}_j)$  individually. Similarly, the likeness measure in unique space is changed from  $\mathbf{x}_i \cdot \mathbf{x}_j$  to  $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$  in higher measurement space. The dot product of  $\Phi(\mathbf{x}_i)$  and  $\Phi(\mathbf{x}_j)$  is known as the kernel function and is represented by  $K(\mathbf{x}_i, \mathbf{x}_j)$ . As the kernel trick assumes that the dot products are comparative in both the spaces, it helps in registering the kernel function in the changed space utilizing the original attribute set. Henceforth the first nonlinear decision boundary mathematical statement in lower measurement space is changed to a comparison of linear decision boundary in higher dimensional space given by:

$$\mathbf{w} \cdot \Phi(\mathbf{x}) + b = 0 \quad (1.18)$$

### 1.4.2 Neural Networks

Neural Networks (NNs) have been effectively used in numerous fields: control field, speech recognition, medical diagnosis, signal and image processing, etc. The main advantages of NNs include self-adaptivity, self-organization, real time operation, and so on. This model takes us an alternate methodology to critical thinking from that of ordinary machines. A NN is comprised of a set of artificial neurons which are called nodes, and they have associations called weight between them. The most straightforward structural planning of fake neural systems is single-layered system, likewise called Perceptron, where inputs connect directly to the outputs through a single layer of weights. The most usually utilized type of NN is the Multi-layer Perceptron (MLP), see Figure 1.4. NNs offer a compelling and exceptionally general structure for speaking to non-linear mapping from a few information variables to a few yield variables [15].

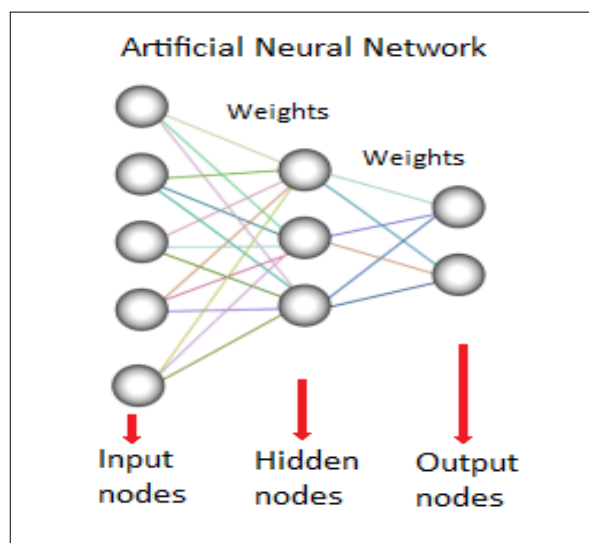


Figure 1.4: NN Multi-layer Perceptron (MLP) [15]

In artificial neural network (ANN), there are several ways to updating the weights associated with the connections between the layers. Most involve initializing the weights and are fed through the network. The error made by the network at the output is then calculated and fed backwards through a process called "backpropagation". This process is then used to update the weights, and by repeated use of this process, the network can learn to distinguish between several different classes. The exact equations involved vary from case to case. More detail will be discussed in section 3.1.9.

INPUT/OUTPUT UNITS, where each connection has a WEIGHT associated with it is given by the following transformation:

$$y_{out} = F(x, W)$$

where  $W$  is the matrix of all weight vectors.

The input can be raw input data or the output of other perceptrons. The output can be the final result (e.g. 1 means yes, 0 means no) or it can be inputs to other perceptrons.

### 1.4.3 K-Nearest Neighbor (KNN)

KNN is a case based classifier. Classification of unknown samples is focused around the separation capacity. This sort of classification is executed by the area of the nearest neighbors in the instance space. Classifying the unknown samples is carried out by naming it with one of the names of the nearest neighbors. The estimation of  $K$  is the quantity of nearest neighbors that ought to be considered to group the obscure example.

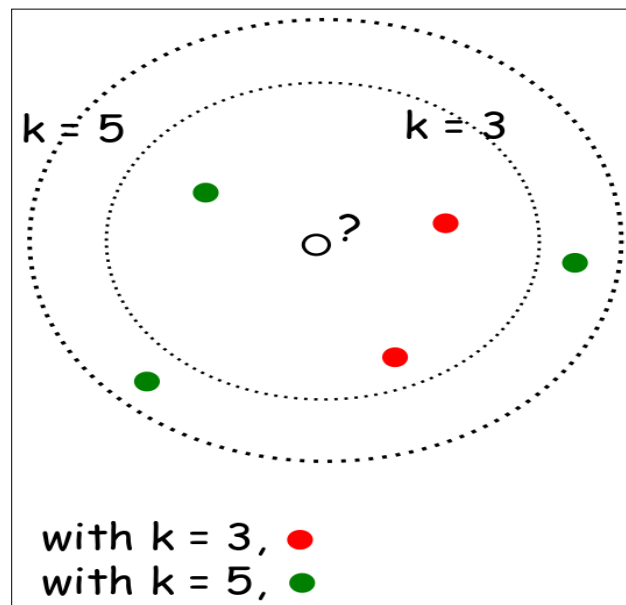


Figure 1.5: working of KNN classifier [47]

Figure 1.5 is utilized to clarify the situation of KNN arrangement. It considers examples fitting in with two classes to be specific, diseased and normal. The unknown sample in the center needs to be classified either diseased sample or normal sample. The inward loop speaks to  $K=3$  and the unknown sample has two neighbors from normal and one from diseased. In view of the greater part voting, the unknown sample will be classified as normal. On account of the external ring  $K = 5$  and the unknown sample has two neighbors from normal and three from diseased and the majority voting will classify the unknown sample to be diseased. The nearest neighbor for an unknown sample can be found by figuring the distance function. Euclidean separation is the ordinarily utilized separation capacity. The usually utilized estimations of  $k$  are 3 and 5. At the point when the estimation of  $k$  is excessively vast, the execution of the classifier may diminish [17]. More detail will be discussed in section 3.1.10.

## **1.5 Objectives of the study and outline of the thesis**

The specific objectives of the study were to:

- Determine the confidence level in classifying an unknown gene sample based on microarray data using SVM comparing with two other classifiers.
- Analyze two different kernels of SVM "linear kernels and the radial kernel" and determine the kernel and the parameters best suited for classification of a given microarray data.
- Use different percentages of distribution of the dataset to training and testing datasets to ensure what should be positive dataset where we are receiving best result.
- Use 10-fold cross validation (LOOCV method) and L1 L2 regularization technique to solve the over- fitting issues and feature selection in classification.
- Use Confusion matrix and ROC curve used to evaluate the performance

The rest of this thesis is organized as follows

- 1) Chapter 2 presents the literature review and some of the previous work done on the microarray classification using SVM, NN and KNN.
- 2) Chapter 3 focuses on the datasets used in thesis, the process followed and application of the process model to the various datasets selected.
- 3) Chapter 4 presents the results obtained from the analyses performed and discusses the results obtained.
- 4) Chapter 5 concludes the report by presenting the observations derived from the current study.

## Chapter 2

### Literature Review

This section primarily analyses the prior works on the classification of microarray data utilizing the support vector machine (SVM), K-nearest neighbor (KNN), and neural networks (NN) classifiers. Prior works involve enhancing the classifiers for improved classification precisions.

S. Mukherjee et al in [18] rendered classification on leukemia cancer data [1] utilizing SVMs. The project dissected classification competence of SVM over the high dimensional microarray data. They rendered the feature selection method propounded by Golub et al in [2]. They analyzed the features and selected top 49, 99 and 999 genes in order to meet the classification need. Classification of all the 7129 genes in dataset was rendered. The study propounded two different procedures; 1) SVM classification without negations; 2) SVM classification with negations. The former method was categorizing the dataset with the use of linear SVM classifier with the topmost 49, 99 and 999 genes and also making use of the other set of 7129 genes. The SVM classifier returned with better accuracy in comparison to the method proposed by Golub et al. The non-linear polynomial kernel SVM classifier didn't enhance precision for the dataset. The second method made use of a confidence threshold value in order to negate the test samples if they reach near the boundary plane. Confidence threshold was measured using Bayesian formulation. Proximity of the training samples to the decision boundary was measured on the basis of leave-one-out fundamental. The allocation function judgment of the proximities was procured by using the non-parametric density estimation algorithm. Then the actual level for the classifier was procured by deducting the estimate from unity. Proximity between the test sample and the decision boundary was measured and if it resulted in less confidence height of the classifier then the verdict is negated and quality of the test sample cannot be analyzed. The general precision was 100% with some samples disallowed in every class of the filtered genes. Of the top 49 genes, 4 were not allowed. Similarly, 2 genes were disallowed from the topmost 99 genes, no rejection from the topmost 999 genes and 3 genes were cancelled out of 7129 genes. They maintained that linear SVM classifier with rejections formulated on confidence values accomplished better on the leukemia cancer dataset [18].

The study performed by Terrence S. Furey et al in [19] based on SVM classification technique in order to segregate the microarray data and also render validation of the cancer tissue sample. The experimental dataset was of 31 samples. These constituted of cancerous ovarian tissues and normal ovarian tissues along with normal non-ovarian tissues. They intended to segregate cancerous tissues from the normal tissues (which consisted of normal ovarian and normal non-ovarian). Mostly the machine learning algorithms underperform for bigger number of features, whereas SVM can easily handle huge dimensional data. Therefore in the study entire dataset was used for bifurcation. The process consisted of bifurcation of the entire dataset using hold-one-out techniques. Then the features were arranged and the topmost features were eventually used for bifurcation. These features were arranged on bases of the scores, which can be measured as a ratio. The numerator of the ratio signifies the gap between the mean expression value of genes in normal tissues and tumor tissues. The denominator is calculated as the total of the standard deviation of normal tissues in addition with the tumor tissues. Linear kernel was used for classification. Then topmost genes on the basis of their scores are procured to train the SVM and unfamiliar samples are bifurcated. For the ovarian dataset two samples (N039 and HWB3) were wrongly classified time and again. They studied these samples by measuring the margin value, which they concluded as the proximity of the sample from the decision boundary. The margin value for misjudged sample N039 was comparatively huge interpreting that it may be labeled wrong. A precise study by the biologist cleared the doubt and indicated that the sample was marked wrong. Another misjudged sample HWBC3 was decided to be an outlier. Also topmost genes procured from feature selection, three out of five genes are associated with cancer. They decided that feature selection can be rendered to ascertain genes that are associated with cancer but implied that it is not 100 percent accurate as some genes that are not associated with cancer were also ranked incorrectly. To simplify the method they diligently analyzed leukemia dataset (Golub et al, 1999) and colon cancer dataset (Alon et al, 1999). The results were kindred to the prior results. They ascertained that SVM could be used for bifurcation of microarray data and results. They further realized that SVM could be useful in classification of microarray data and for studying the misjudged specimen. It has been thought out that with the use of non-linear kernel, bifurcation precision might be exalted for complicated dataset, which are generally arduous to bifurcate with the use of a simple linear SVM kernel [19].

The study organized by Sung-Huai Hsieh et al [13] stressed on segregation of leukemia dataset with the use of support vector machines. They offered a SVM classifier using Information Gain (IG) as feature selection method. Information gain is the reduction in the entropy when the data is partitioned based on that gene. Entropy can be useful for assessing if a feature is beneficial in carrying out the bifurcation, in addition to defining the relation amongst the training dataset. Microarray dataset was sub-divided further into two independent sets – training and test data. Then training dataset undergoes feature selection by computing the IG values for the genes. Genes with higher rank IG values were chosen for segregation. Further the training data is prepared to take care of any outliers and reduce the learning model bias. This training data is then used to prepare SVM classifier. Radial-based function (RBF) kernel with grid searching was chosen in SVM model and accurate values for the parameters of RBF (penalty C and gamma g) were concluded. This model was put to test by arranging cross validation method and also by an independent test dataset. The paper propounded that SVM classifier model with IG feature selection rendered sufficient precision (98.10%) [13].

John Phan et al. debated regarding several parameter optimization techniques for SVM classifier so as to enhance segregation [14]. The study stressed on distinguishing genetic markers that can separate renal cell carcinoma (RCC) subtypes. The dataset constituted of different types including 13 clear-cell RCC, 4 chromophobe and 3 oncocytoma. The study strived to differentiate the clear cell RCC from the other two categories. Initially with the use of SVM, genes were ranked according to their capacity to differentiate classes. Then the prognostic precision was computed considering leave one out method. Sequential minimal optimization technique was put to use for enhancing SVM. It has been debated that precision developed by the SVM is largely dependent on the kernel selected and the parameters. The study stressed on linear and radial basis kernel (RBF). The linear and RBF kernels have parameter 'C' that correlate to the penalty for incorrect classification. Higher intensity of 'C' means more penalties, making the classification model to be over-fitting. Conversely, smaller value of 'C' results in a more relaxed model that may find it tough to bifurcate the unfamiliar data correctly. In the paper, 'C' was varied from 0.01 to 10 for linear and 0.01 to 100 for the RBF kernel and thus the highest value of 'C' was defined along with computing the prediction rate. On the basis of the average prediction rate collected by altering the parameters they propounded highest values for 'C' as 0.1 for linear kernel and 1 for RBF. Also the results indicated that the computed value of sigma resulted in



optimal average prediction rate. Similarly, the best value of sigma for the RBF kernel was propounded as the mean of closest 2m neighbors to the data point where m corresponds to the dimensionality of the data point. Also, the results indicated that the computed value of sigma obtained highest average prediction rate. It has been identified that the genes picked by the use of those specific processes were believed to have biological pertinence on the basis of gene ontology and literature [14].

Mihir Sewak [16] organized a study on bifurcation of leukemia dataset with the use of committee neural networks. In the study, a panel of neural networks was built to bifurcate samples of leukemia. Two types of classifications were propounded namely, binary classification and three class classification. For binary classification, the samples were divided into Acute Lymphoblastic Leukemia and Acute Myeloid Leukemia. For three class classification, the panel distinguished Acute Lymphoblastic Leukemia further into sub categories. Dataset was exposed to different preprocessing steps which resulted in obtaining more informative genes. Further these genes were put to use training various neural networks. The excelling neural networks were chosen over the others and selected into the committee. Part of the original dataset helped validating the committee and, the committee produced a staggering precision of 100% for binary classification and 97% for three class classification.

Huilin Xiang et. al. [20] study stresses on the kernel based distance metric learning classification for microarray data. Data was classified using the maneuvered KNN method. This method constituted the kernel optimization for binary and multi class data. Then the metric distance procured by kernel optimization method exalted the ability of separating the class of data in feature space.

The chosen method enhanced the workability of KNN classifier for the bifurcation of gene expression data. Smaller training data had rendered the algorithm unfit for working. To surpass this issue they had chosen an affected re-sampling method to enhance the size of the training data sample. This altered KNN method bifurcated leukemia dataset with an optimal precision of 96% and it was comparable with the workability of SVM classifier.

Manik Dhawan [17] propounded the study on leukemia cancer dataset through committee approach. KNN classifier was put to use for bifurcation. The dataset had 48 ALL and 25 AML

samples. The dataset was randomized by choosing a validation set from the main dataset and used the remaining dataset to select the committee of classifier to classify the validation set. Random selection of 10 ALL and 10 AML samples were chosen for forming Training and testing datasets.

Training and testing datasets were preprocessed and the dataset values were normalized between -1 to 1. Further these training and testing datasets were bifurcated into 5 different groups. The first group constituted 50 topmost genes and the secondary group had the next subsequent top 50 genes and so on. Each training and testing groups were granted to four classifiers KNN (3,2), KNN (5,3), KNN (7,4), KNN (9,5). The most optimal working classifier from every group was selected into the committee. The committee was brought to power by the use of validation sets. 100% precision was achieved for 7 out of 12 validation sets which assured the soundness of the committee.

## Chapter 3

### Materials and Methods

The goal of this study is to determine the confidence level in classifying an unknown gene sample based on microarray data using SVM and comparing the results with two other classifiers. The study also provides analysis of two different kernels of SVM, namely “linear kernels and the radial kernel.” Further this research focuses on determining the kernel and the parameters best suited for classification of given microarray data. The analysis was performed using different percentages of distribution of the dataset for training and testing datasets to ensure the best positive dataset that gives the best results. Ten-fold cross validation (LOOCV method) and the L1, L2 regularization techniques are processed to solve the over-fitting issues and feature selection in classification. A confusion matrix and ROC curve is used to evaluate the performance.

#### 3.1 Methodology

##### 3.1.1 Dataset selection

Four different microarray datasets are used in the study: i) Colon cancer dataset by Alon et al. obtained from Princeton University Gene Expression Project. [23]; ii) Leukemia cancer dataset by Golub et al, obtained from Broad Institute [22]; iii) Breast cancer dataset by van’t Veer et al, obtained from NCBI [24]; iv) Prostate cancer dataset by Singh et al. [25].

##### Dataset of Colon Cancer

An Affymetrix Hum6000 array of oligonucleotide was used for the purpose of analyzing the 62 samples, including 20 normal samples and 40 tumor samples, collected from patients of colon-cancer. Expression of 2000 genes along with the maximum lower strength across all the 62 tissues was included in the matrix I2000. The genes were arranged in the descending order of lowest strength. Every entry in I2000 is gene intensity derived from around 20 particular pairs of the gene present on the chip and is received through filtering. In this study, 40 samples which are marked as negative have been collected from tumors and 22 samples, marked as positive have

been collected from biopsies of the healthy part of the colon of the same individuals. A total of two thousand genes were picked from 6500 depending on the measured level of expression [23]. The format of the dataset is shown in table 3.1.

**Table 3.1: Format of Colon cancer dataset**

	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	
H55933	8589.4163	9164.2537	3825.705	6246.4487	3230.3287	.....
R39465	5468.2409	6719.5295	6970.3614	7823.5341	3694.45	.....
R39465	4263.4075	4883.4487	5369.9688	5955.835	3400.74	.....
R85482	4064.9357	3718.1589	4705.65	3975.5643	3463.5857	.....
U14973	1997.8929	2015.2214	1166.5536	2002.6131	2181.4202	.....
R02593	5282.325	5569.9071	1572.1679	2130.5429	2922.7821	.....
T51496	2169.72	3849.0588	1325.4025	1531.1425	2069.2463	.....
H80240	2773.4212	2793.3875	1472.2587	1714.6312	2948.575	.....
.....	.....	.....	.....	.....	.....	.....

#### **Dataset of Leukemia**

High density oligonucleotide based microarrays were used for creating Leukemia dataset. These microarrays were made by Affymetrix and includes profiles of gene expression for two samples, ALL and AML collected from tumors. The received dataset was already segregated into test data and training data. The total dataset has 25 AML samples and 48 ALL samples; and for each of the sample there are a total of 7129 genes. Each of the samples used in the experiment included two columns linked by the microarray dataset. The level of expression of the gene in the microarray experiment is presented in the first column. The second column, represented by CALL determines whether the expression value is due to the gene or due to noise. It might take

any of the three values: presence, marginal or absence, represented by P, M and A according to the signal [22]. The format of the dataset is shown in table 3.2

**Table 3.2: Format of Leukemia cancer dataset**

Accession Number	Sample 1	CALL	Sample 2	CALL	Sample 3	
A28102_at	151	A	484	A	118	.....
AB000114_at	72	A	61	A	16	.....
AB000115_at	281	A	118	A	197	.....
AB000220_at	36	A	39	A	39	.....
AB000381_s_at	29	P	38	A	50	.....
AB000409_at	-299	A	-11	A	237	.....
AB000410_s_at	-336	A	-116	P	-129	.....
AB000449_at	57	P	274	P	311	.....
.....	.....	.....	.....	.....	.....	.....

### Dataset of Breast Cancer

Breast cancer microarray study performed by van't Veer et al. released this gene expression data [24]. Seventy eight (78) primary samples of breast cancer were collected from patients over 55 years of age with lymph node negative; 34 samples out of these 78 were from patients who within 5 years had grown distant metastases, and the rest 44 were from patients who were free from the ailments after at least a period of 5 years. Utilizing the fluorescent dye reversal technique, two hybridizations are done for each tumor on the microarrays. And the microarrays are containing almost 25000 human genes. Inkjet oligonucleotide method synthesized these genes. Level of gene expression is numbered as 24188. Few missing values are included in this

data set. Depending on the correlations between the gene expressions, remaining missing values are estimated. This thesis focused on 4948 number of genes from 24188 genes in 78 tumor samples. Table 3.3 shows the formatted dataset.

**Table 3.3: Format of Breast cancer dataset**

Probe Set ID	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	...
Contig45645_RC	-0.125	-0.27	-0.141	-0.149	-0.382	...
Contig44916_RC	0.07	0.123	0.025	0.068	0.064	...
D25272	-0.006	0.056	-0.031	-0.084	0.033	...
J00129	-0.575	-0.499	-0.465	-0.557	-0.873	...
Contig29982_RC	-0.585	-0.402	-0.533	-0.595	-0.474	...
.....	....	...	....	...	...	...

#### **Dataset of the Prostate cancer**

Singh et al. (2002) present the Gene expression data containing 6033 genes for 102 samples from the microarray analysis and measurements of the individual gene expression. 102 samples include 50 healthy men and 52 patients of prostate cancer. Healthy samples are marked as “healthy” and diseased samples are labeled as “cancer” [25]. The format of the dataset is shown in table 3.4.

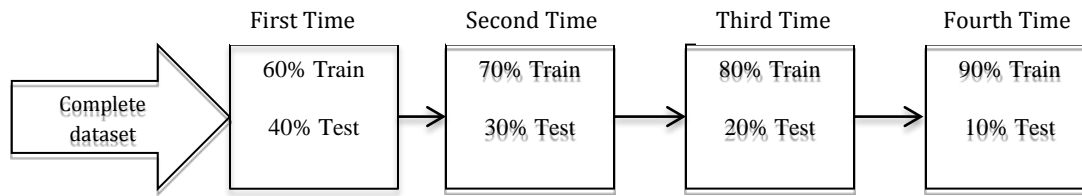
**Table3.4: Format of Prostate cancer dataset**

	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	....
AFFX-MurIL2_at	-9	-2	-6	0	-1	....
AFFX-MurIL10_at	1	1	17	9	0	....
AFFX-MurIL4_at	15	4	29	19	5	....
AFFX-MurFAS_at	-2	-2	4	-10	0	....
AFFX-BioB-5_at	-3	-5	-11	-18	-4	....
AFFX-BioB-M_at	4	0	-8	-18	1	....
AFFX-BioB-3_at	8	8	10	5	6	....
AFFX-BioC-5_at	-12	-5	-24	-33	-4	....
AFFX-BioC-3_at	-12	-9	-32	-31	-9	....
AFFX-BioDn-5_at	20	7	-20	14	12	....
AFFX-BioDn-3_at	-6	-4	-11	-12	-5	....
AFFX-CreX-5_at	0	0	3	-2	0	....
....	....	....	....	....	....	....

### 3.1.2 Overview of the Procedure

An overview of the procedure adopted in the study is given in the Figure 3.2. The following steps describe the procedure:

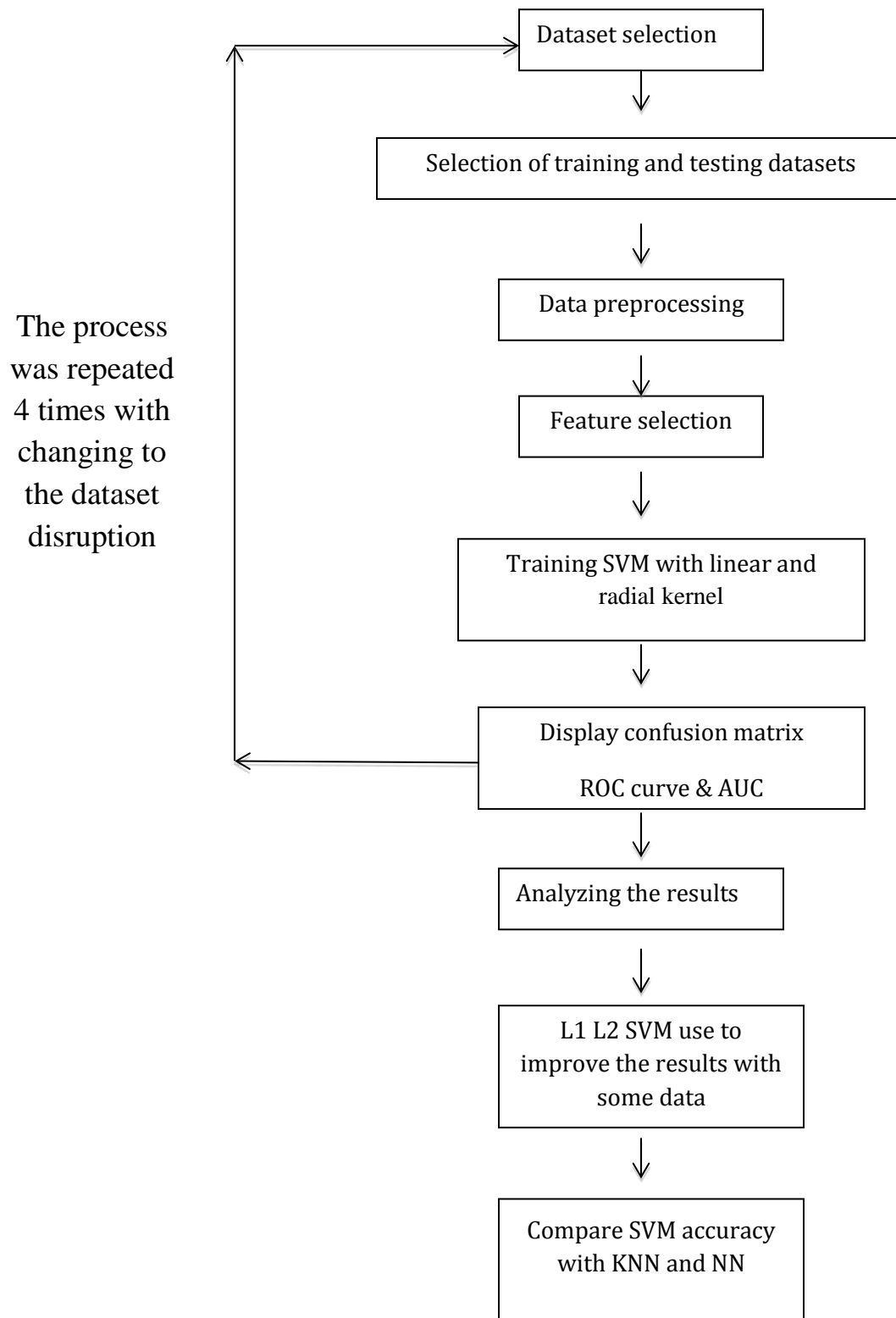
- 1- The complete dataset was used to generate training and testing dataset randomly.
- 2- Four different splits for the data samples were used for training dataset and test dataset respectively.



**Figure 3.3.a Block diagram of Selection training and testing**

- 3- Training datasets were subjected to preprocessing which helped in the removal of genes that did not contribute to the classification.
- 4- After preprocessing the data, feature selection was performed and filter method used to choose the informative genes.
- 5- Samples in test dataset were classified using SVM with two different Kernels with 10-fold cross validation.
- 6- The test error was obtained between linear kernel and radial kernel, and linear kernel was chosen.
- 7- The accuracy of classification was calculated and is displayed by confusion matrix.
- 8- ROC curve and AUC (area under the curve) was calculated to evaluate the classifier.
- 9- All steps were repeated four times with different split for the dataset each time with different percentage.
- 10- After analyzing the results, L1 and L2 regularization with SVM was used to improve the accuracy with some dataset.

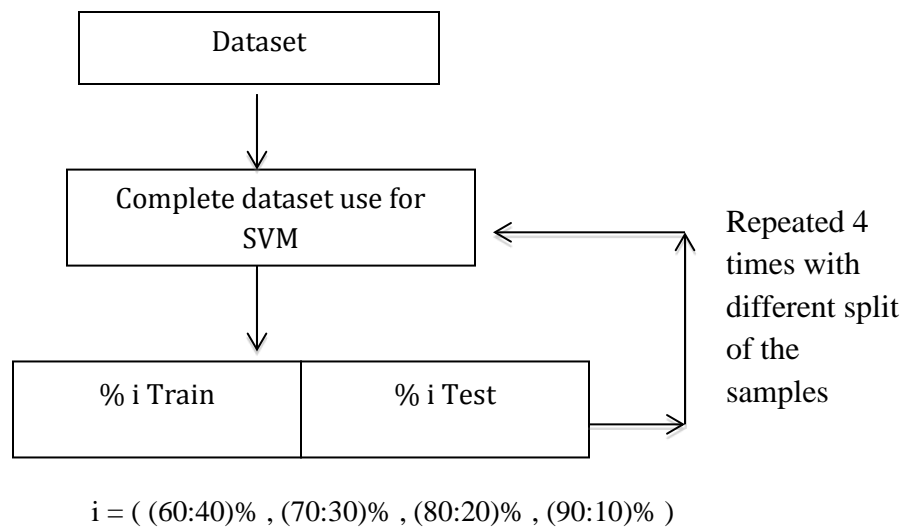




**Figure 3.2 Protocol used for the current study**

### 3.1.3 Selection of training and testing dataset

The process followed for the dataset selection is shown in Figure 3.3. For each of the 4 datasets, the complete dataset was used to generate training and test data randomly. Different percentages of distribution of the dataset were chosen. The distribution selected were ((60:40)%, (70:30)%, (80:20)%, (90:10)%) for training dataset and test dataset respectively. For generating test data in first distribution, 40% samples from each category were picked from the entire dataset. The rest of the 60% samples in the dataset formed the training dataset. In second distribution, 30% samples for test data and 70% samples for training dataset were selected. In third distribution, 20% samples for test data and 80% samples for training dataset were selected. In fourth distribution, 10% samples for test data and 90% samples for training dataset were selected. ROC graph and area under the curve representing exactly what should be positive dataset shows the best results among different split ratios. The process of generating the dataset was performed in R program and the R code is presented in Appendix A. The training and test data sample distribution for each of the 4 data sets is shown in Table 3.5.



**Figure 3.3: Selection of training and testing datasets**

**Table 3.5: Sample's distribution of training and test data for 4 datasets**

dataset		% 60: %40			% 70: %30			% 80: %20			% 90: %10		
Leukemia	Train	All	AML	Total	All	AML	Total	All	AML	Total	All	AML	Total
		13	31	44	36	15	51	40	18	58	44	21	65
	Test	16	12	28	11	10	21	7	7	14	3	4	7
Colon	Train	Tumor	Normal	Total	Tumor	Normal	Total	Tumor	Normal	Total	Tumor	Normal	Total
		26	12	38	29	15	44	34	16	50	37	19	56
	Test	14	10	24	11	7	18	6	6	12	3	3	6
Breast	Train	DM	NODM	Total	DM	NODM	Total	DM	NODM	Total	DM	NODM	Total
		20	27	47	24	31	55	28	35	63	31	40	71
	Test	14	17	31	10	13	23	6	9	15	3	4	7
Prostate	Train	Healthy	Cancer	Total	Healthy	Cancer	Total	Healthy	Cancer	Total	Healthy	Cancer	Total
		31	31	62	37	35	72	42	40	82	44	48	92
	Test	19	21	40	12	18	30	8	12	20	6	4	10

### 3.1.3 Preprocessing of Data

Preprocessing was done over the full datasets. Steps are as follows:

- Preprocessing involved refurbishing the dataset by eliminating the housekeeping genes namely, the endogenous control genes. Housekeeping genes are responsible for basic cellular activities. Expression of these genes remains the same in every cell with minor changes. There is no role of these genes in classification. Therefore it is better to remove them from dataset [26].
- In the leukemia dataset following the analysis of Adarsh Jose et al [27], genes having CALL values labeled as (A) or Absent in comparison with (P) or present were eliminated. Like the breast cancer, there is no mention about the CALL value in prostate as well as colon cancer. In those cases, all the genes were considered for further processing.
- Expression values are affected by the background errors and it is reflected in the extreme values. Such values are known as outliers. Values above 16,000 cannot be measured by the imaging equipment. Below 20 values are also caused from the background noise [28, 29]. For dealing with these outliers, there is a need of performing fundamental transformations. The data are winsorized by the transformation keeping the minimal value around 100 and maximum value around 16000.

### 3.1.4 Feature selection

The feature selection methods fall into two categories filter methods and wrapper methods [31]. Filter was applied to the test dataset genes to know which genes are most relevant to the binary classification task and these genes were selected to remove noisy or irrelevant genes. R program for preprocessing, transformations and filtering the data is presented in Appendix B.

### 3.1.5 Training the SVM

One of the most reliable and standard data mining and machine learning tool was Support Vector Machines or (SVMs) [32]. Numbers of effective and impactful applications have been implied for classifying the microarray gene expression data. In order to use SVMs for classifying the gene expression data, let us assume  $M$  is the provided vectors of gene expression, and kernel function is denoted by  $K$  as shown in Equation 3.1. More importantly, it is frequently chosen as a polynomial of degree  $d$ .

$$K(M, m_i) = (m^T m_i + 1)^d \quad (3.1)$$

Following is the discriminant function

$$L(M) = \sum_{i=1}^T \alpha_i c_i K(M, m_i) \quad (3.2)$$

Where  $m_{i=1}^T$  is a set of gene vectors and  $c_{i=1}^T$  is the corresponding class,  $\alpha_i$  is the weight of training sample  $y_i$ . It signifies the strength of the sample with which it is embedded in the function of final decision. Only a part of the training vectors will be associated with a non-zero  $\alpha_i$ . These vectors are so-called support vectors. In order to increase the gap between the samples from two respective classes, training process needs to update the weights.

$$F(a) = \sum_{i=1}^T \alpha_i (2 - \beta_i L(m_i)) = 2 \sum_{i=1}^T \alpha_i - \sum_{i=1}^T \sum_{j=1}^T \alpha_i \alpha_j \beta_i \beta_j K(y_i, y_j) \quad (3.3)$$

where  $\alpha_i \geq 0$ .

In this study preprocessed training data was utilized for training the model of the SVM classifier. R implementation of SVM classifier in e1071 package was utilized for training the SVM model. Two different kernels were utilized to train the SVM model, linear kernel and Gaussian radial basis function (RBF). Linear kernel function maps the examples in the training data onto a feature space and decides the ideal maximal margin hyper-plane that partitions both classes of data. The function utilized for linear kernel is

$$K(X_i, X_j) = X_i \cdot X_j^T \quad (3.4)$$

Where " $\cdot$ " implies the dot product between the two vectors. In a Gaussian RBF kernel the data samples are changed to a high dimensional space most likely to an infinite dimensional space

where the data belonging to two categories can be separated utilizing a linear hyper-plane. The kernel function utilized for RBF is:

$$K(X_i, X_j) = \exp\left(-\gamma \|X_i - X_j\|^2\right), \gamma > 0 \quad (3.5)$$

$\gamma$  is the kernel parameter. A default estimation of 1 was picked for  $\gamma$ . As aforementioned, deciding an ideal hyper-plane is an optimization concern. Quadratic programming optimization technique was utilized for deciding the separating hyper-plane. The SVM classifier was trained utilizing the data and the known classes of the training data. Information given to the SVM classifier was changed by varying the kernel function utilized by the SVM classifier. Test error was computed for every Kernel capacity with 10- fold cross validation. According to the test error results, linear kernel was selected. Appendix C shows training the SVM classifier with Linear Kernel and radial Kernel.

### 3.1.5 Linear Kernel SVM

After choosing the kernel to use, the training error is computed by applying the model to the data on which it is trained. R code is presented in Appendix A.

```
> predicted <- predict(svm1, Xm)
```

*Predict* function used to choose the best model on test set. When we use a linear kernel, there are no kernel parameters to specify. The only value we have to choose is the misclassification cost. Default is cost=1, but another value achieves better cross-validation results [34]. We tune the SVM by trying different values of cost with 10-fold cross-validation. We employ cross-validation to see what the cross-validated training set error rate is.

```
ranges = list(cost=c(0.001, 0.01, 0.05, 0.1, 1, 5, 10, 100)))
```

the *cost* list contain the value of misclassification cost.

Once we know the best parameter setting, we can use it to learn an SVM. These two steps (finding best parameter and then training an SVM) are combined in the function *tune.out*, which returns the SVM with the best parameters.

```
> tune.out = tune(svm, y~., data=df.train, kernel="linear", decision.values=TRUE, ranges =
```

```
list(cost=c(0.001, 0.01, 0.05, 0.1, 1, 5, 10, 100)))
```

Then we choose the best model from 10 –fold cross validation

```
>bestmod = tune.out$best.model
```

*summary* function use to display detailed of performance results.

```
>summary(bestmod)
```

Next we use the best model on the training set to predict the class of samples in the test set.

```
ypred = predict(bestmod, df.test, decision.values=TRUE)
```

*bestmod* to use the best model to predict on test set,*df.test* test set.

```
( tbl = table(predict=ypred, truth=df.test$y) )
```

### 3.1.6 Evaluated methods

#### *Confusion Matrix*

After we use the best model to predict on test dataset, a confusion matrix is used for evaluating the classifier according to their performance. Confusion matrix is termed as error matrix. It is a tabulated presentation that reveals algorithm performance. It is basically a supervised learning method. Confusion matrix shows the information regarding predicted and actual classifications performed following a classification system. Instance of the predicted class is reflected by the columns while the actual class is reflected by the rows. This name originated from the concept that whether the system confuses two classes or not.

Suppose we use our classifier on a test set of labeled tuples. P is the number of positive tuples and N is the number of negative tuples. For each tuple, we compare the classifier's class label prediction with the tuple's known class label. The following definitions are used for evaluation measures [39].

**True positives (TP):** These refer to the positive tuples that were correctly labeled by the classifier. Let TP be the number of true positives.

**True negatives (TN):** These are the negative tuples that were correctly labeled by the classifier. Let TN be the number of true negatives.

**False positives (FP):** These are the negative tuples that were incorrectly labeled as positive. Let FP be the number of false positives.

**False negatives (FN):** These are the positive tuples that were incorrectly labeled as negative. Let FN be the number of false negatives.

Table 3.6 shows the confusion matrix, which is used to analyze how well the classifier can recognize tuples of different classes.

**Table 3.6: Confusion matrix**

		Predicted Class		
Actual class		Yes	No	Total
	Yes	TP	FN	P
	No	FP	TN	N
	Total	P'	N'	P+N

Several evaluation measures have been defined for the confusion matrix.

- The *accuracy* of a classifier on a given test set is the percentage of test tuples that are correctly classified by the classifier. It is given as:  $accuracy = \frac{TP+TN}{P+N}$
- The *error rate* or *misclassification rate* of a classifier M is  $1 - accuracy(M)$  and is computed as:  $error\ rate = \frac{FP+FN}{P+N}$
- *Sensitivity* is the *true positive* rate, i.e., the proportion of positive tuples that are correctly identified and is given as:  $sensitivity = \frac{TP}{P}$
- *Specificity* is the *true negative* rate, i.e., the proportion of negative tuples that are correctly identified and is given as:  $specificity = \frac{TN}{N}$



- *Precision* is a measure of *exactness*, i.e., percentage of tuples labeled as positive and is given as:  $precision = \frac{TP}{TP+FP}$
- *Recall* is a measure of *completeness*, i.e., percentage of positive tuples and is the same as sensitivity or the *true positive rate* and is given as:  $recall = \frac{TP}{TP+FN} = \frac{TP}{P}$

R code for computing the confusion matrix is presented in Appendix D.

### ***ROC curve***

In order to observe the performance of classifiers, we use ROC curve. ROC graph is accompanied with data mining for observing the performance of classifiers, for comparing them and is also useful in other cases where:

- There are unequal class distributions
- Different costs of misclassification

False positive rate is plotted on the X-axis and true positive rate is plotted on the Y-axis in the ROC graph. In this graph perfect classifier is (0,1) and it has been found that both positive and negative cases are classified by this classifier. In (0,1), 1 stands for true positive rate and 0 stands for false positive rate. Besides this, (0,0) point signifies all negative cases, whereas (1,1) represents every positive case. In all the classification, point (1,0) is found as incorrect. Several cases are like that where parameter is adjustable for increasing TP at an increased cost of FP or even decreased cost of FP while TP is also decreasing. Pair of (FP, TP) is provided by each parameter setting and for plotting ROC curve, it has an effective application. Once we got the area inside the curve, we did comparison study with other classifiers. R code for plotting ROC curve and AUC (area under curve) is presented in Appendix E.

### **3.1.7 L1, L2-Regularization**

In the field of machine learning problem regularization is defined as a method in which extra information is introduced to avoid over-fitting [40]. This information is added in a restrictive format discouraging non-essential complexity, an example of which is setting a minimum

specification for smoothness or bounds on the vector space norm. The main use of regularization in machine learning is to help in the process of model selection keeping in mind that over-fitting has to be discouraged by rejecting any model that has violated the set specifications. L1 and L2 regularization are the most common variants in machine learning. These can be added to learning algorithms that minimize a loss function

$$E(X, Y) \quad (3.6)$$

by instead minimizing

$$E(X, Y) + \alpha \|w\| \quad (3.7)$$

Here  $w$  has been assigned the value of model's weight vector.  $\|\cdot\|$  is either the L1 norm or the squared L2 norm, and  $\alpha$  is a free parameter that needs to be tuned empirically (typically by cross-validation). The method of regularization finds its use in various fields like binary and multiclass logistic regression, neural nets, support vector machines, conditional random fields and matrix decomposition methods. If regularization is being used in neural nets then L2 regularization is known as "weight decay". Amongst L1 and L2 regularization it is L1 that is more commonly used because it is more successful in delivering sparse models thus enabling feature selection to be done within the framework of the learning algorithm but a drawback of L1 is that it is not differentiable so it may need to be changed in learning algorithm especially algorithms which cater to gradient based learners.

### 3.1.8 SVM with L1, L2 Regularization

Analysis has proved that support vector mechanism (SVM) is a good classification and regression method, which efficiently uses machine-learning theories resulting in maximum possible predictive accuracy thus eliminating chances of data over-fitting. Standard SVM usually employs L2 as its regularization method. Good results are obtained if the data set is free of noise. In case the data set deals with many noise variables L1 regularization SVM is a better choice. In data training SVM algorithms, the penalty functions are already inbuilt so at times they give excellent performance but in other cases might yield unsatisfactory results. The weight of vector  $w$  is minimized by connection between weighted SVM's and L1- SVMs

$$\underset{w, b, \xi}{\text{minimize}} \quad \frac{1}{2} \|w\|_1 + C \sum_i \xi_i \quad (3.8)$$

subject to 
$$y_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i, \quad \xi_i \geq 0 \quad \forall i. \quad (3.9)$$

A wrapper package known as R package (LiblineaR) is used in this study. LiblineaR is actually a linear data classifier, which has a database consisting of millions of examples and features. Both L2 regularization and L1 regularization can work in LiblineaR. The standout feature of LiblineaR is multi-class classification, cross-validation for model selection, probability estimates (logistic regression) or weights for unbalanced data. R code for SVM with L1, L2 Regularization is presented in Appendix F.

### 3.1.9 Neural Networks

Artificial Neural Networks (ANN's) are computational models inspired by biological neural networks (the central nervous systems of animals, in particular the brain). These computation models help in studying functions that depend on analysis of many undefined inputs. Artificial neural networks are generally presented as systems of interconnected "neurons" which can compute values from inputs, and are capable of machine learning as well as pattern recognition thanks to their adaptive nature. The simplest example of 'NN' is a single layered network known as Perceptron which has a direct connection between inputs and outputs by a series of weights. Multi layered Perceptron (MLP) is the most widely used type of NN (Figure 1.4). Through the use of NNs an efficient and wide based framework is created for linear mapping of many input variables producing many output variables. In a NN each input is multiplied by a connection weight, which is denoted by  $W_n$ . Usually, the resulting products are added (Eq. 3.10) and entered into a transfer activation function to produce an output.

$$Y_{output} = f(\sum_{n=0}^N X_n W_n) \quad (3.10)$$

Equation 3.11 depicts logistic sigmoid function, which is the most commonly used activation function.

$$g(a) = \frac{1}{1 + \exp(-a)} \quad (3.11)$$

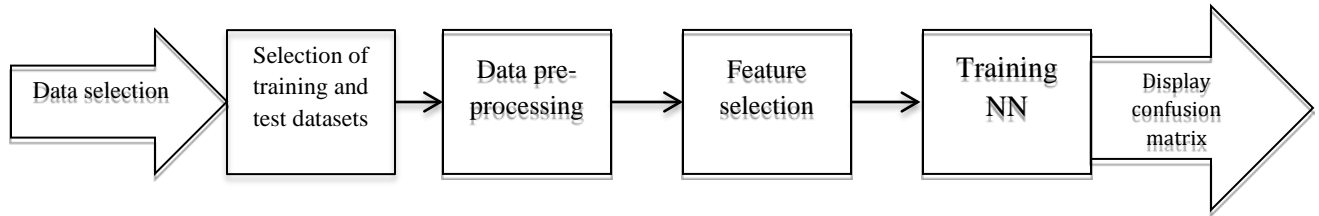
The outputs that are generated lie between 0 and 1 and the inputs vary from negative to positive infinity. If the final result to be achieved is a trained network then the weights of each unit should be such that the difference between the actual and desired output is minimized.

Practically, backpropagation algorithm (BP) is the most commonly used supervised learning algorithm. BP algorithm functions on the basis of a gradient descent technique where the gradient of errors functions is the deciding factor of the proportion of weight change.

$$E = \frac{1}{2} \sum_0^n (z_0 - t_0)^2 \quad (3.12)$$

$$W_{ij}(t + 1) = W_{ij}(t) - \eta \frac{\partial E}{\partial W_{ij}} + \beta(W_{ij}(t) - W_{ij}(t - 1)) \quad (3.13)$$

The symbols denote the following values  $Z_0$  –real output from final layer node,  $t_0$  depicts desired output,  $W_{ij}(t)$  stands for the weight from node  $i$  to  $j$  during  $t$ th iteration and actually denotes the two constants of learning rate and momentum rate. Gene expression data obtained by using ANN is being introduced based on sample filtering. Verification of the model can be done by simulation tests that utilized data sets from four microarrays and comparison was made with SVM and k nearest neighbor.



**Figure 3.4 Block diagram of NN**

The data used as input in NN classifier to program the model was the same preprocessed training data that was used with SVM. NN classifier's R implementation (nnet) was used for NN model training. After the training and test dataset has been generated, the first step to be performed is the normalization of predicted value. Then the attribute of the dataset that is to be predicted is used.

```
gdf<-data.frame (ALL=colonCA_trainSub$class,t(exprs(colonCa_trainSub)))
```

*colonCA\_trainSub\$class* this subset from train dataset. Then the training data is entered into the model.

```
nn1<-nnet(y, data=df.train,size=5,decay=0.01,Max NWts= 6131)
```

Then the test set is predicted

```
Print (table(predict(nn1,new=df.test,type="class"),y.test)))
```

The final step is to calculate classification accuracy and plot area under the curve. Appendix G shows the R code for all NN procedures.

### 3.1.10 K-Nearest Neighbor (KNN)

One of the simplest classifications used at present is the K Nearest Neighbors (KNN). Bressan and Vitri were the first to use it in 1951 and it has undergone many modifications since then and new forms have emerged like the probabilistic nearest neighbor model (Holmes, 2002). There is no need of a training phase in the KNN method and the new sample's class is taken as the most common class. In this case, decision function is determined by

$$L(M) = \sum_{i=1}^T \alpha_i c_i K(M, m_i) \quad (3.14)$$

where K denotes the set of neighbor closest to the new point x. The time complexity of this method is O(N), where N is the number of training samples. The advantage KNN has over SVM is that it can deal with multi –class classification with the new point's class decided by its neighbors. A distance metric like Euclidean Distance (ED) (Eq.3.15) or the Cosine Coefficient (CC) (Eq.3.16) between two gene expression vectors is used to define the set of nearest neighbors.

$$ED = \sqrt{\sum (m_i - m_j)^2} \quad (3.15)$$

$$CC = \frac{\sum m_i m_j}{\sqrt{\sum m_i^2 \sum m_j^2}} \quad (3.16)$$

In this, gene expression vectors are denoted by  $m_i$  and  $m_j$ . It is always better to define the influence of each neighbor based on its distance from the new point. This can be done by multiplying class levels by a weighting term.

In this study, KNN algorithm was applied to all genes in the four microarray datasets and each gene's category was derived using other genes as the training sample. Classification accuracy

was confirmed by setting up different k values. Error rate has been calculated by randomly split cross validation method. The samples were split from the data available for testing and the rest for training dataset. Appendix H shows the R code for all K-NN Procedures.

### **3.1.11 Obtaining and analyzing the results**

Regularization technique was performed with support vector machine classification for sorting out over-fitting issues and enabling accurate feature selection. Accuracy values of SVM classifier, NN and K-NN were compared. The process included training the classifier, calculating the confidence levels and testing with an independent database. According to the results obtained the best classification model for a microarray data set was obtained. The Chapter 4 discusses the detailed results.

## **Chapter 4**

### **Results and Discussion**

Four different datasets were used to study the classification methods' performance and to find a classifier with the best performance compared to the other classifiers. The SVM classifier was trained with the pertinent data; the kernel and the parameters best suited for classification of a given microarray data were determined. Different distributions of the dataset for training and test datasets were used to ensure what should be positive dataset. The 10-fold cross validation and regularization technique was processed to solve the over-fitting issues and feature selection. The confusion matrix and the area under curve were calculated. We trained neural network classifiers and k- nearest neighbor classifiers with the same datasets. The following section presents a detailed explanation of the results.

#### **4.1 Colon cancer dataset**

In this section we used the microarray data set of colon cancer to train the classifiers by examining each method with suitable parameters. The original dataset consisted of 40 tumor samples and 22 normal samples. We evaluated the performance of each classifier by its ability to classify to the same number of samples, and then chose the best one.

##### **4.1.1 Training the SVM with the colon cancer dataset**

As described in the previous chapter, this SVM classifier is a kernel technique. The first step in the process is obtaining the test error for linear Kernel and radial kernel to choose the better Kernel function with less error. The test error was 22.7% for the linear kernel and 27.3% for the radial kernel. According to the test error result linear kernel was chosen. The cost of misclassification was then chosen and tried with the 10-fold cross validation to return the SVM with the best parameters. Then the best training set was used to predict the class of samples in the test set. The processes were repeated four times with different splits of the dataset for the training and test datasets. The Roc curve and area under curve were calculated to compare the classifier performance with different splits for the dataset. Figures 4.1.a through 4.1.d displays

the ROC curve for SVM classifiers for each distribution of the dataset to training dataset and test datasets.

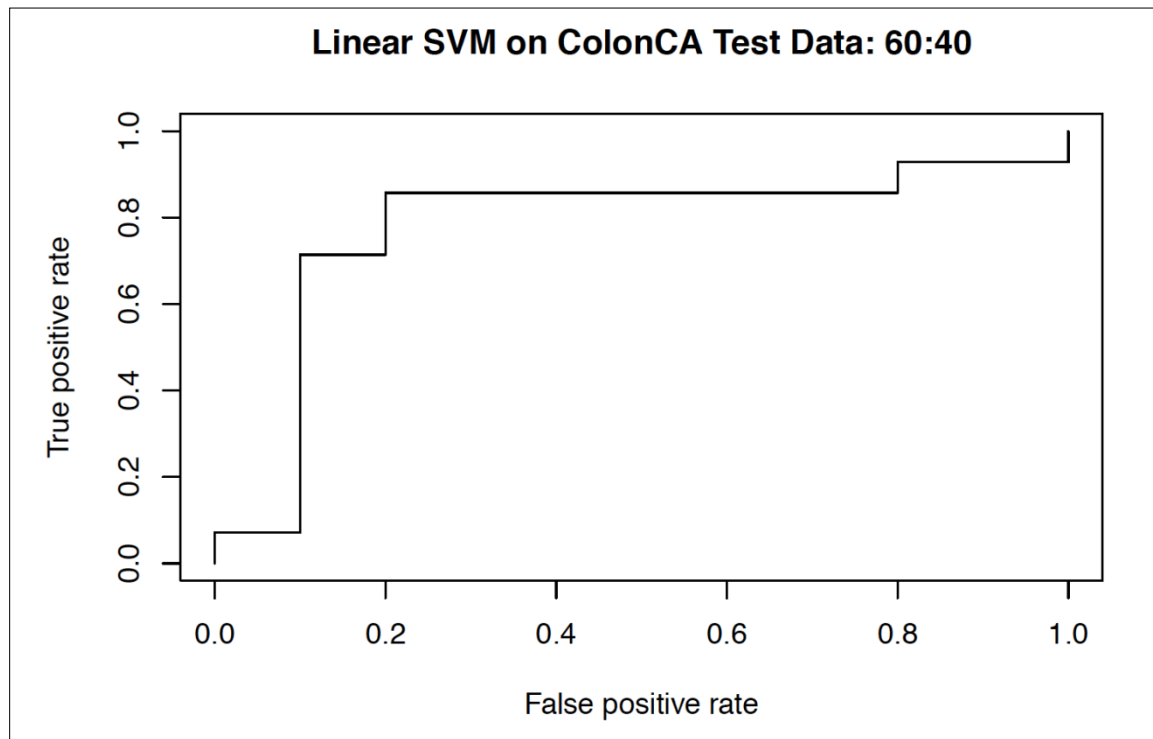


Figure 4.1.a: Linear SVM: colon dataset (60:40)

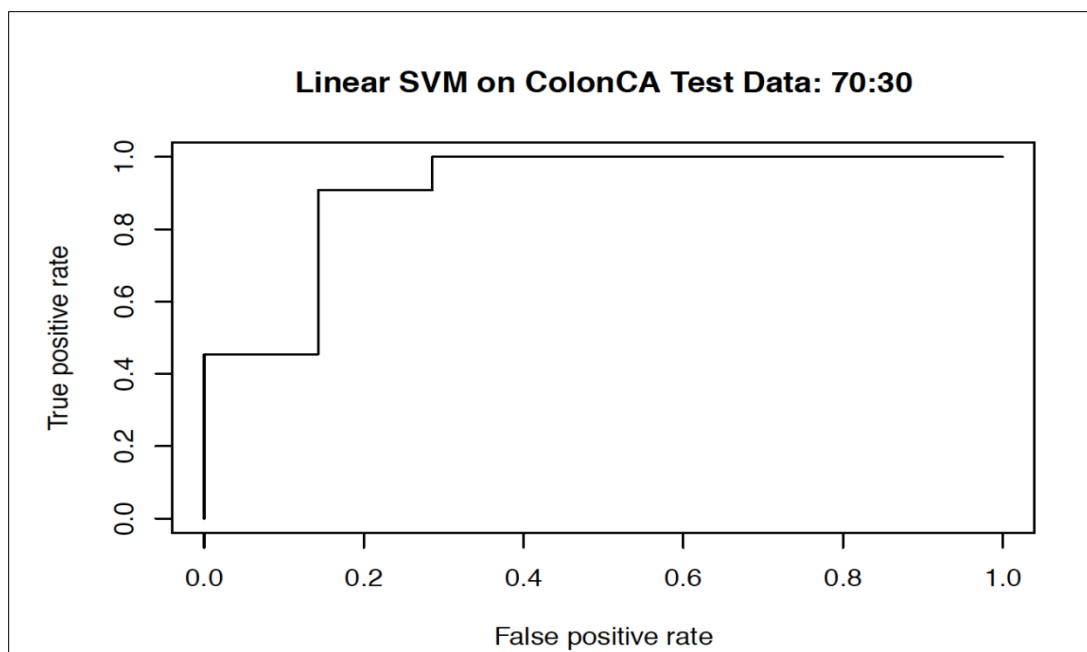
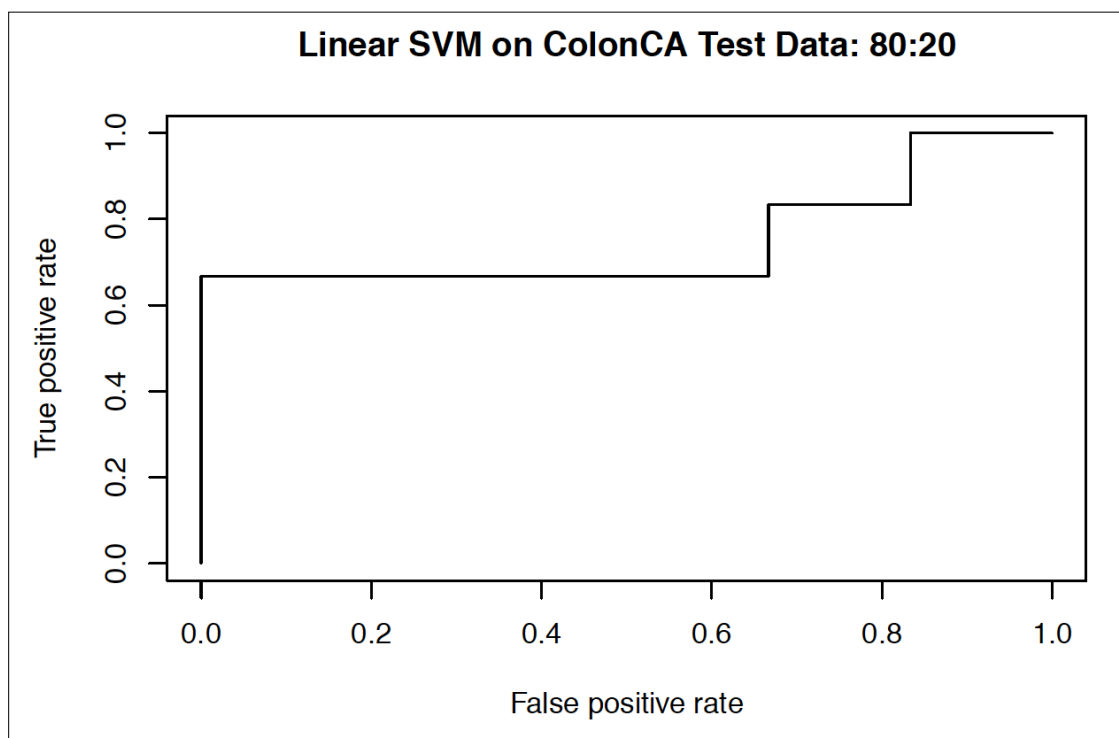
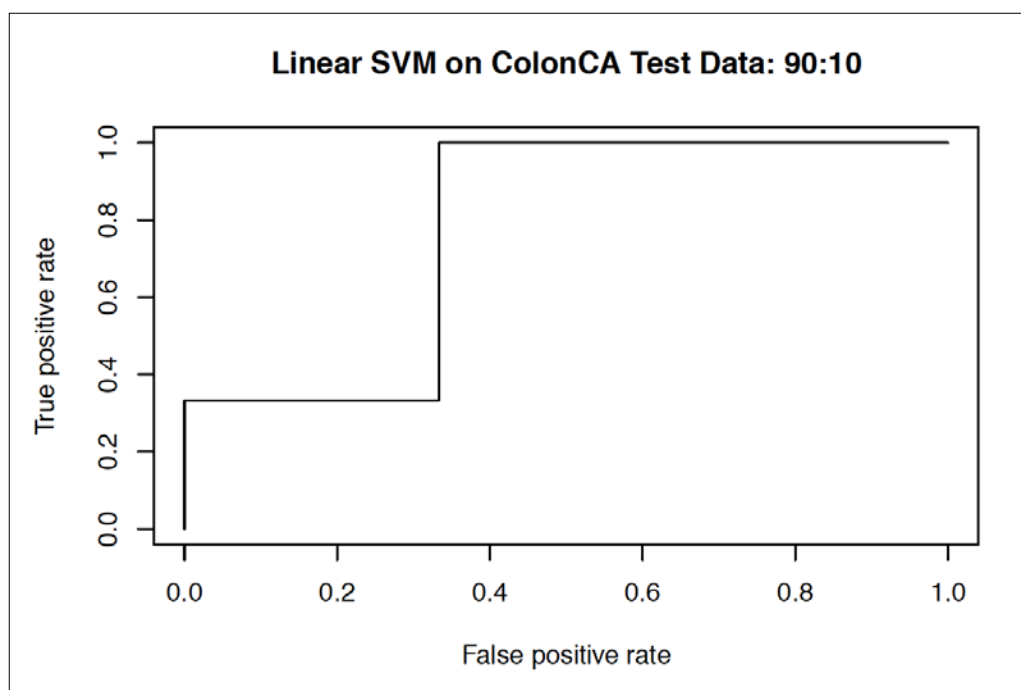


Figure 4.1.b: Linear SVM: colon dataset (70:30)





**Figure 4.1.c: Linear SVM: colon dataset (80:20)**



**Figure 4.1.d: Linear SVM: colon dataset (90:10)**

For each ROC curve we calculated the area under curve. As shown in Fig. 4.1.a the area under curve for the distribution of 60% for training and 40% for test dataset was 0.778. Fig. 4.1.b displays the ROC curve for the classifier with 70% training and 30% testing; the area under

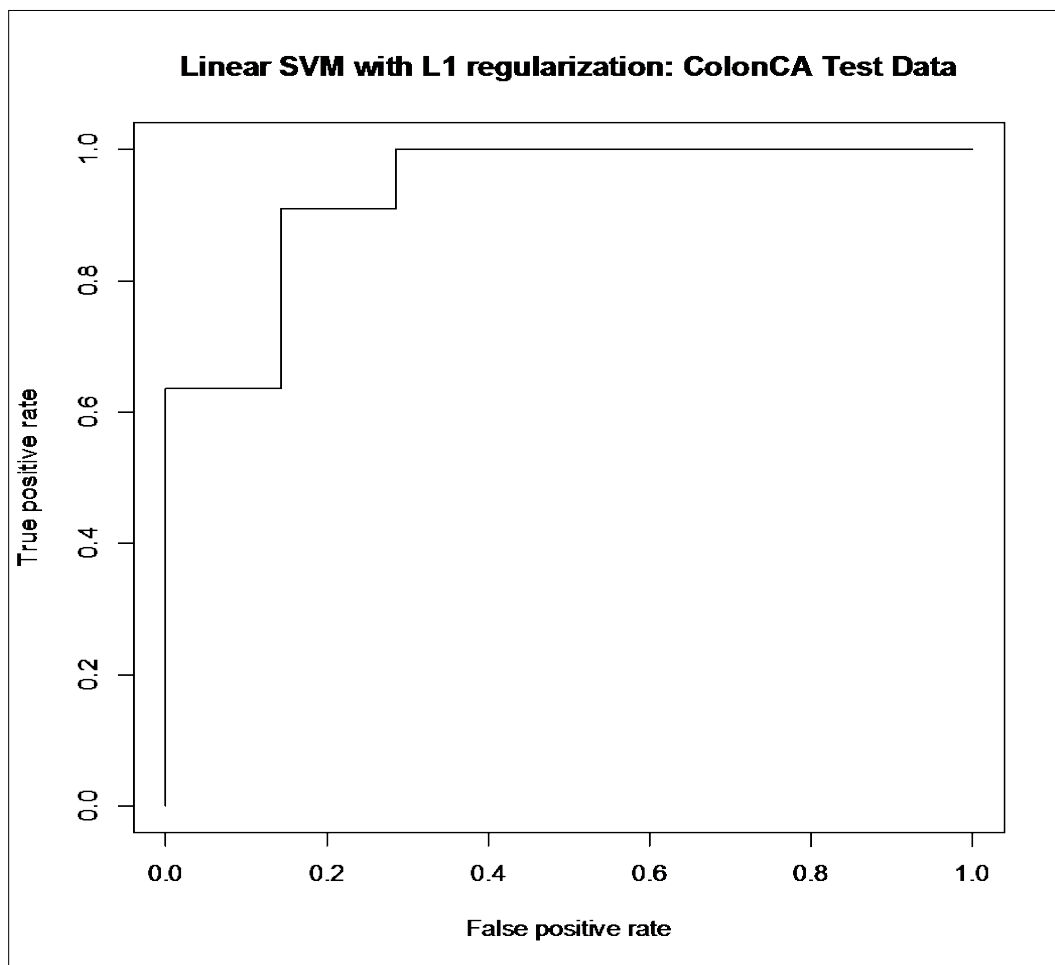
curve (AUC) was 0.90. Fig. 4.1.c displays the ROC curve for the classifier with 80% training and 20% testing; the AUC was 0.75. Finally, Fig. 4.1.c displays the ROC curve for the classifier with 90% training and 10% testing; the AUC was 0.77. When we compared the AUC between the groups of data the distribution of 70% training and 30% testing gave the highest AUC, which is 90%. For this result, the split of (70:30) was chosen to complete the process. After that the confusion matrix display was used to evaluate the performance of the classifier. The confusion matrix shows that two samples out of eleven from the tumor (t) class were misclassified. For the normal (n) class one sample out of seven was misclassified.

		Predicted class		
		n	t	total
Actual class	n	5	1	6
	t	2	10	12
	total	7	11	18

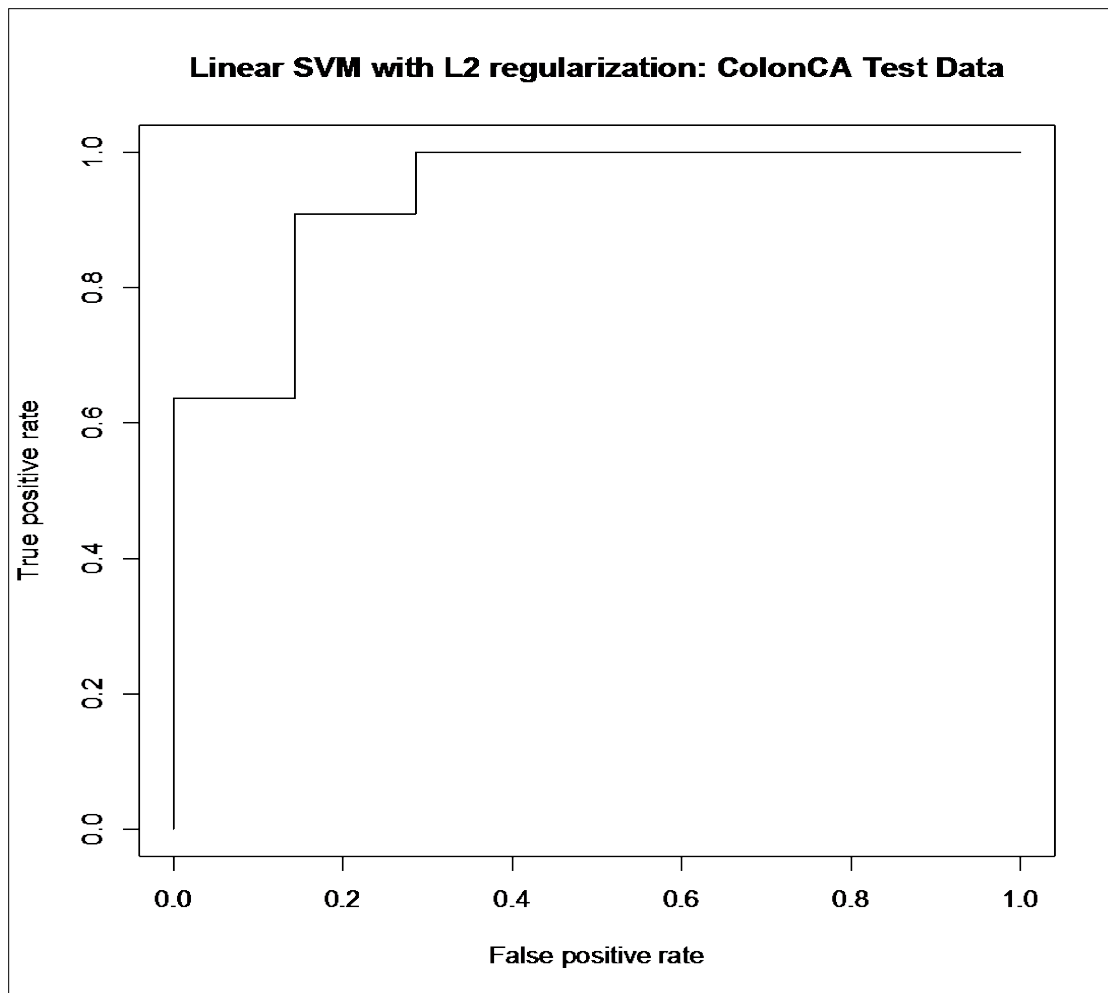
The accuracy for this classifier with colon dataset was 0.833.

#### 4.1.2 SVM with L1, L2- Regularization for the colon cancer dataset

When we trained the SVM classifier with colon cancer data the accuracy of the classifier was 83% with AUC 90%. Then we used L1, L2 regularization with SVM to improve the classifier's performance. We ran the model with 10-fold cross validation, then we re-trained the best model with the best-cost value. Then we used the best model to predict the class of samples in the test set. Figures 4.2.a, 4.2.b displays the ROC curve for SVM with L1 and L2 regularizations.



**Figure 4.2.a: Linear SVM with L1 regularization: colon dataset**



**Figure 4.2.b: Linear SVM with L2 regularization: colon dataset**

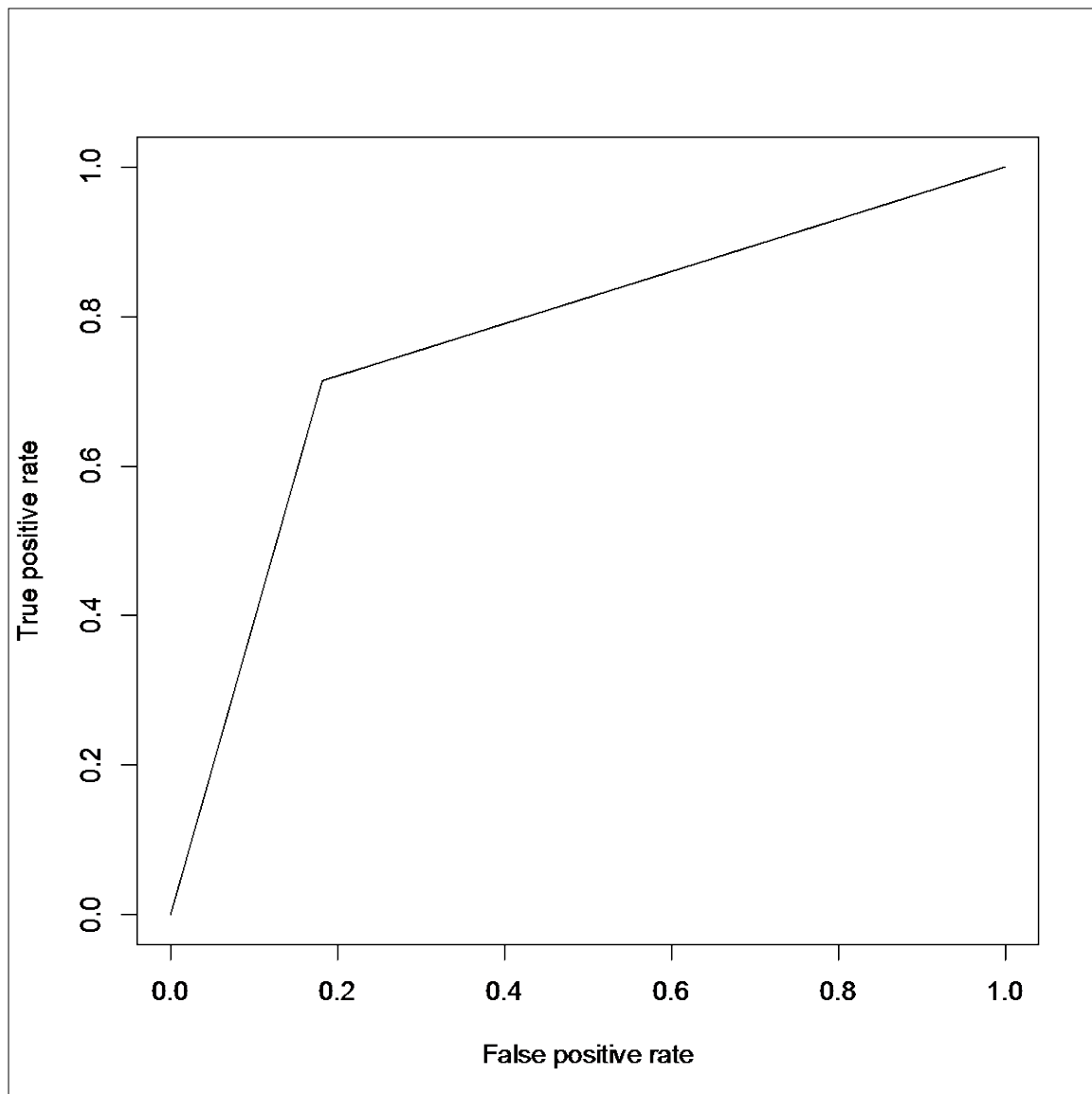
The confusion matrix for SVM with L1, L2 regularization shows three misclassified samples, two from tumor class and one from normal class.

		Predicted class		
		n	t	total
Actual Class	n	6	2	8
	t	1	9	10
	total	7	11	18

The accuracy of the classifier didn't change, which remained 83%. But we improved AUC, which became 93%.

### 4.1.3 Training Neural Network for colon cancer dataset

The second classifier we used in this study was the neural network. We trained this method with the same pre-processed data. Distribution ranged up to 70% for the training dataset and 30% for the test data. We normalized the value to be predicted and then trained an NN for the training data. Finally, we tested the output from the neural network. Figure 4.3 displays Roc curve of the NN for the colon cancer dataset.



**Figure 4.3: Neural Network: colon dataset**

We calculated the area under curve (AUC) to evaluate the classifier, which turned out to be 76%. The confusion matrix showed that two samples out of eleven from the tumor class were misclassified. Two samples out of seven from the normal class were misclassified.

		Predicted class		
Actual Class		n	t	total
	n	5	2	7
	t	2	9	11
	total	7	11	18

The accuracy for this classifier with colon dataset was 0.77.

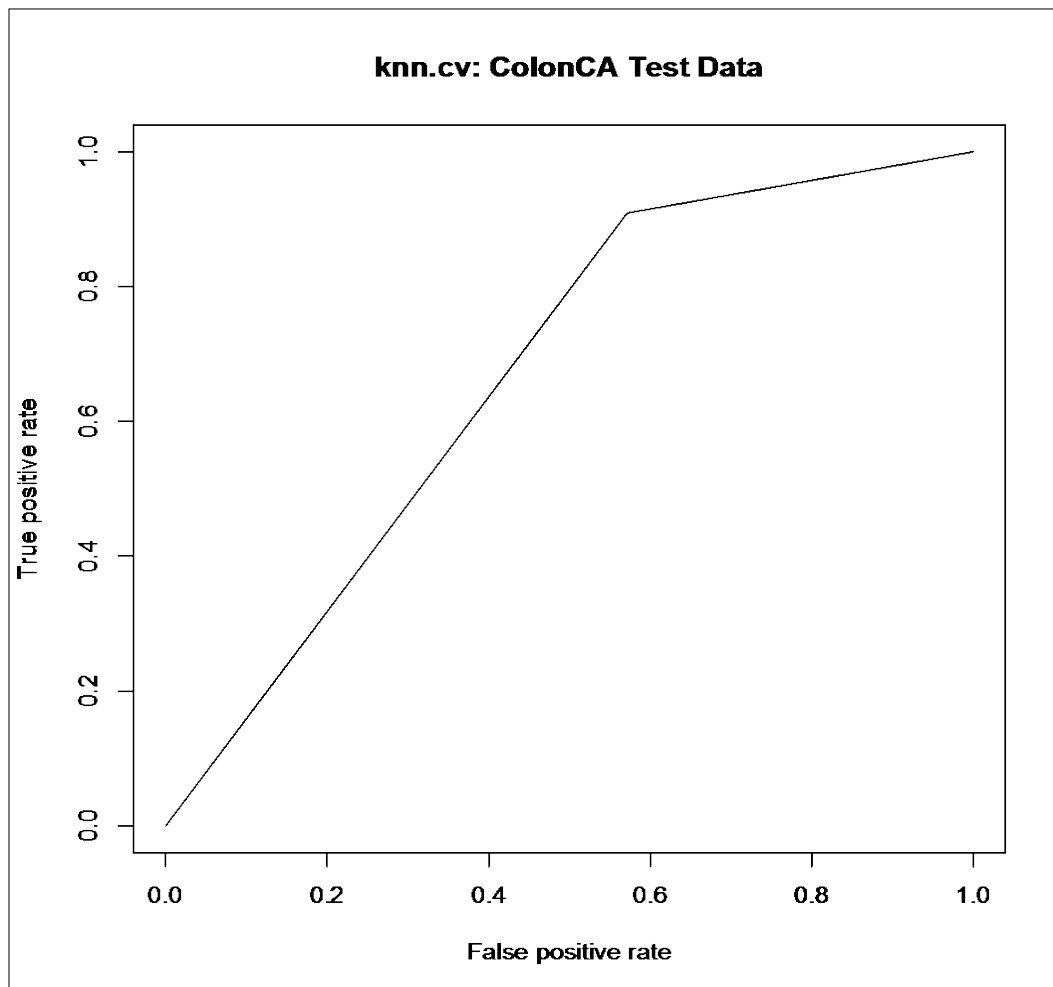
#### 4.1.4 Training K-Nearest Neighbor for colon cancer dataset

The third classification method we used in this study was KNN. Same pre-processed data were used and samples were split into 30% for the test dataset and 70% for the training dataset. We set different k values to choose the best number of neighbors for KNN, which gives the smallest test error rates. We calculated the confusion matrix to evaluate performance of the classifier with each k.

With k=1 the confusion matrix showed that the false positive rate for tumor class three out of seven, and the false positive rate for normal class was three out of eleven. For k=2 the false positive rate for tumor class was four out of seven, and the false positive rate for normal class was two out of eleven. The false positive rate for tumor class with k=3 was four out of seven and one out of eleven for the normal class. Finally, with k=4 the false positive rate for tumor class was five out of seven, and the false positive rate for normal class was two out of eleven. Based on these outputs, k=3 yields the smallest test error rates. Therefore, we chose k=3 as being the best number of neighbors for KNN with this colon dataset.

		Predicted class		
Actual Class		n	t	total
	n	3	1	4
	t	4	10	14
	total	7	11	18

The classification result based on  $k=3$  is shown in the ROC curve plot of Figure 4.4.



**Figure 4.4: K-Nearest Neighbor: colon dataset**

We calculated the area under curve for this classifier, which was 73%. The accuracy for this classifier with the colon dataset was 0.72.

## 4.2 Leukemia cancer dataset

We have used the Leukemia dataset as second group of microarray dataset to examine the performance of our three classifiers. Our goal was to determine the confidence level of classifying two types of leukemia known as ALL and AML. The original dataset consists of 48 ALL samples and 25 AML samples. We trained the three classifiers with this pre-classified data and, according to the test error, the best classification method was determined.

### 4.2.1 Training the SVM with Leukemia cancer dataset

We trained linear kernel SVM to classify the leukemia dataset. The cost of misclassification was chosen and tried with 10-fold cross validation to return the SVM with best parameters. Then the best model of training set was used to predict the class of samples in the test set. We repeated the process four times with different splits of the dataset for training and test dataset. ROC curve and area under curve were calculated to compare the classifier performance with different split for the dataset. Figures 4.5.a, 4.5.b, 4.5.c, and 4.5.d display ROC curve for SVM classifier for each distribution of the training dataset and test dataset.

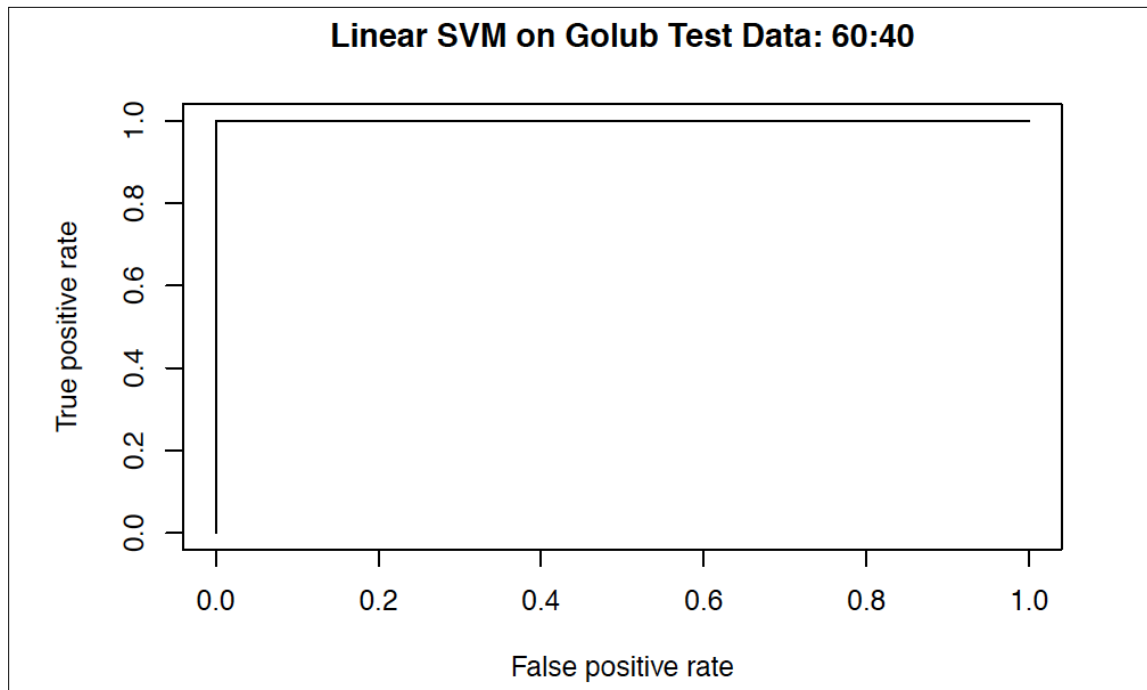
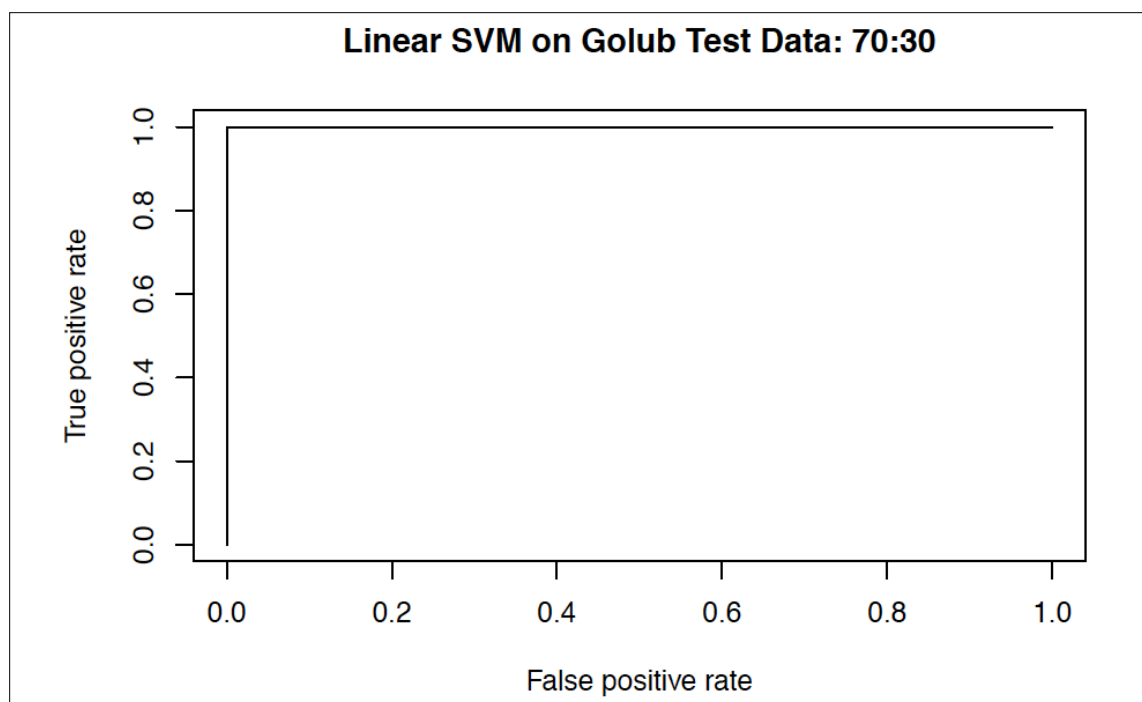
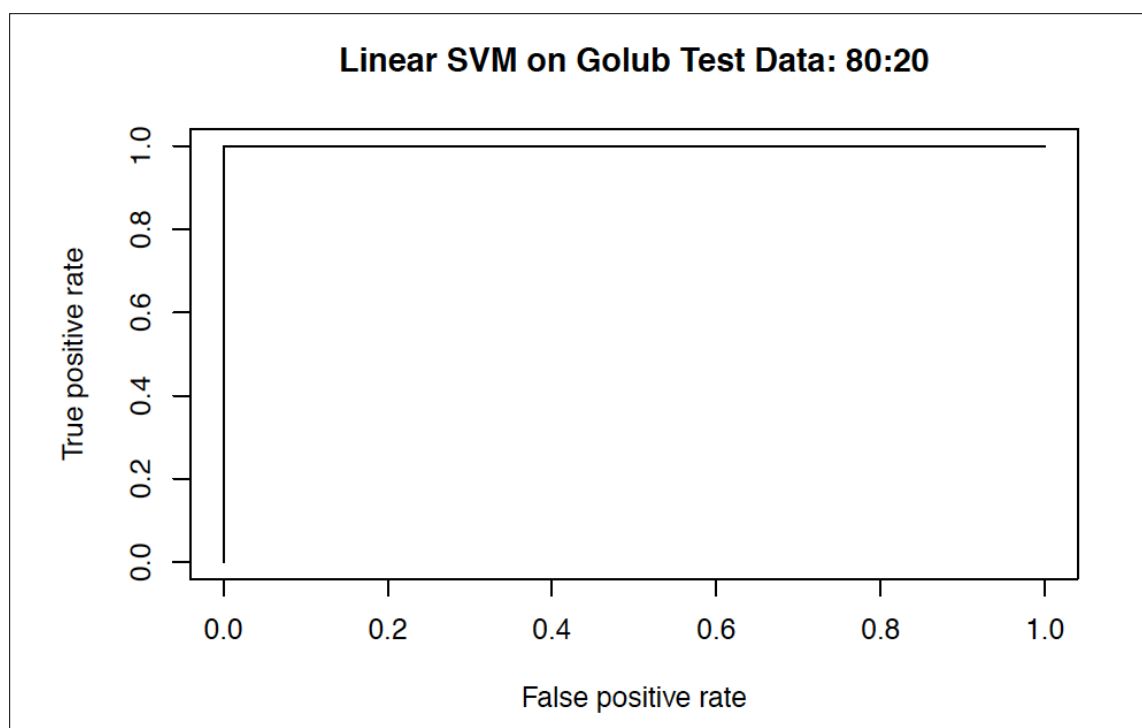


Figure 4.5.a: Linear SVM: Leukemia dataset (60:40)

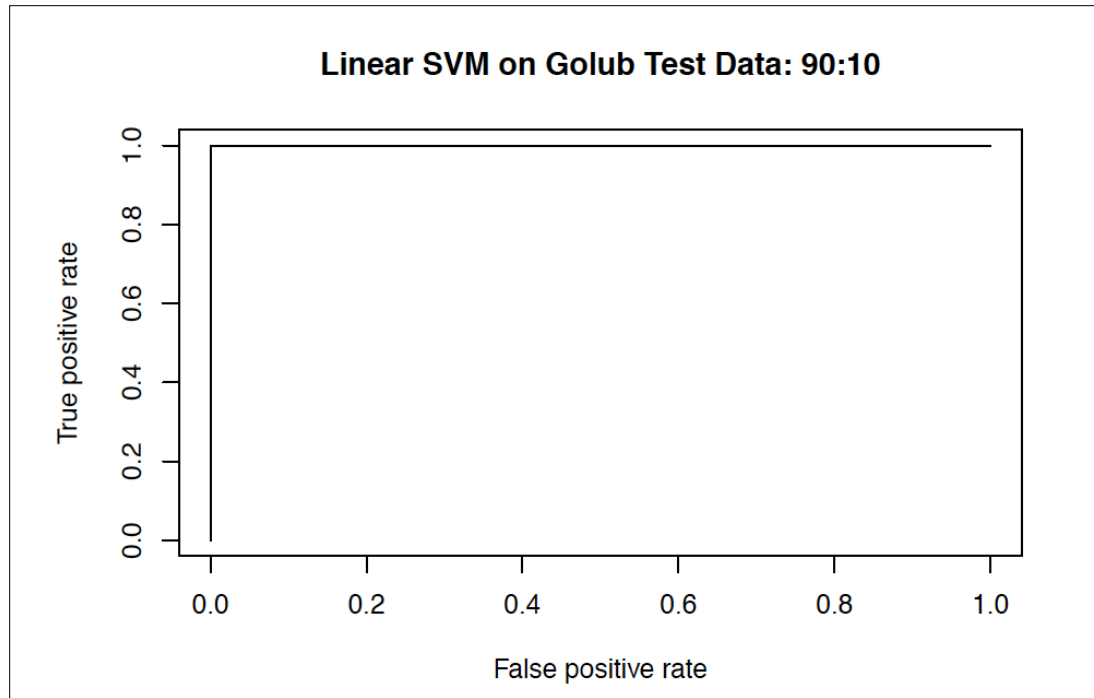




**Figure 4.5.b: Linear SVM: Leukemia dataset (70:30)**



**Figure 4.5.c: Linear SVM: Leukemia dataset (80:20)**



**Figure 4.5.d: Linear SVM: Leukemia dataset (90:10)**

For each ROC curve we calculated the area under curve. As shown in Figures 4.5.a, 4.5.b, 4.5.c, 4.5.d the area under curves for all distribution of the data for training and testing was 100%. This result shows that these datasets were made in good frames, changing partitions does not affect the classification result. We have chosen the split of (70:30) to complete the process. Confusion matrix was used to evaluate performance of the classifier. The confusion matrix shows that one sample out of fourteen from the AML class was misclassified. For the ALL class none were misclassified.

**Predicted class**

<b>Actual Class</b>		ALL	AML	total
	ALL	20	1	21
	AML	0	13	13
	total	20	14	24

The accuracy for this classifier with leukemia dataset was 0.97 with AUC 100%. The SVM classifier performed well with this group of data and gives a high accuracy. According to this

result we did not need the regularization technique to improve the classifier performance with leukemia data.

#### 4.2.2 Training Neural Network for leukemia cancer dataset

We trained neural network classifier with leukemia dataset. After doing the pre-process step the distribution of the data was 70% for training and 30% for testing. We normalized the values to be predicted and trained the NN for the training data. ROC curve shown in Figure 4.6 was used to evaluate the performance of NN for leukemia cancer dataset.

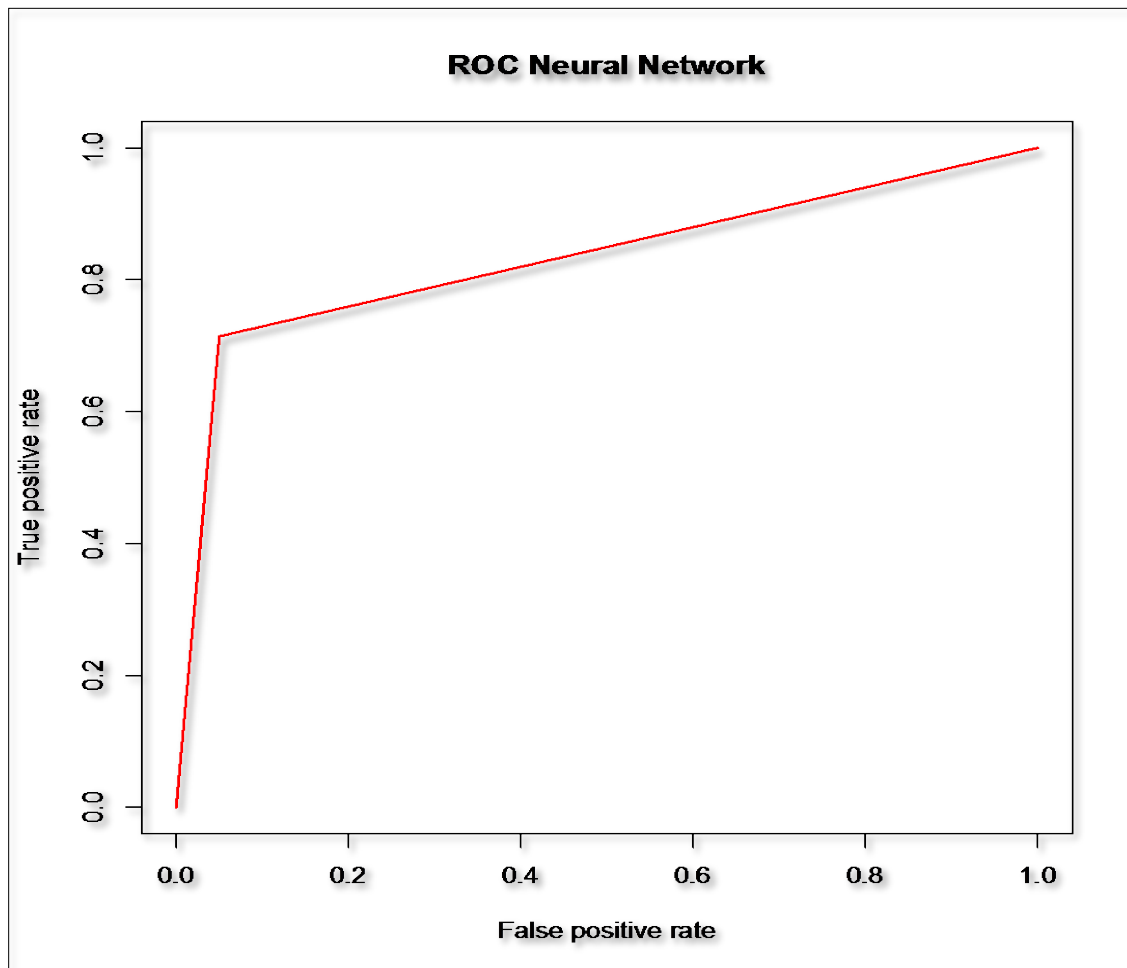


Figure 4.6: Neural Network: Leukemia dataset

We calculated the area under curve for this classifier, which was AUC 86%. The confusion matrix shows that one sample out of twenty samples from the ALL class was misclassified. Four samples out of fourteen samples from the AML class were misclassified.

		Predicted class		
		ALL	AML	total
Actual Class	ALL	19	4	23
	AML	1	10	11
	total	20	14	24

The accuracy for this classifier with leukemia dataset was 0.85.

#### 4.2.3 Training K-Nearest Neighbor for leukemia cancer dataset

We trained K-Nearest Neighbor classifier with the leukemia dataset. The same pre-processed data were used and samples were split into 30% for test dataset and 70% for the training dataset. We set different k values to choose the best number of neighbors for KNN, which gives the smallest test error rates. We calculated the confusion matrix to evaluate performance of the classifier with each k.

With k=1 the confusion matrix shows that the false positive rate for ALL class was zero out of twenty, and the false positive rate for AML class was three out of fourteen. For k=2 the false positive rate for ALL class was one out of twenty, and the false positive rate for AML class was two out of fourteen. The false positive rate for ALL class with k=3 was zero out of twenty and four out of fourteen for the AML class. Finally, with k=4 the false positive rate for ALL class was zero out of twenty, and the false positive rate for AML class was four out of fourteen. Based on these outputs, k=1 yields the smallest test error rates. Therefore, we chose k=1 as the best number of neighbors for KNN with this leukemia dataset.

		Predicted class		
		ALL	AML	total
Actual Class	ALL	20	3	23
	AML	0	11	11
	total	20	14	24

The classification result based on k=1 is shown in the ROC curve plot of Figure 4.7.

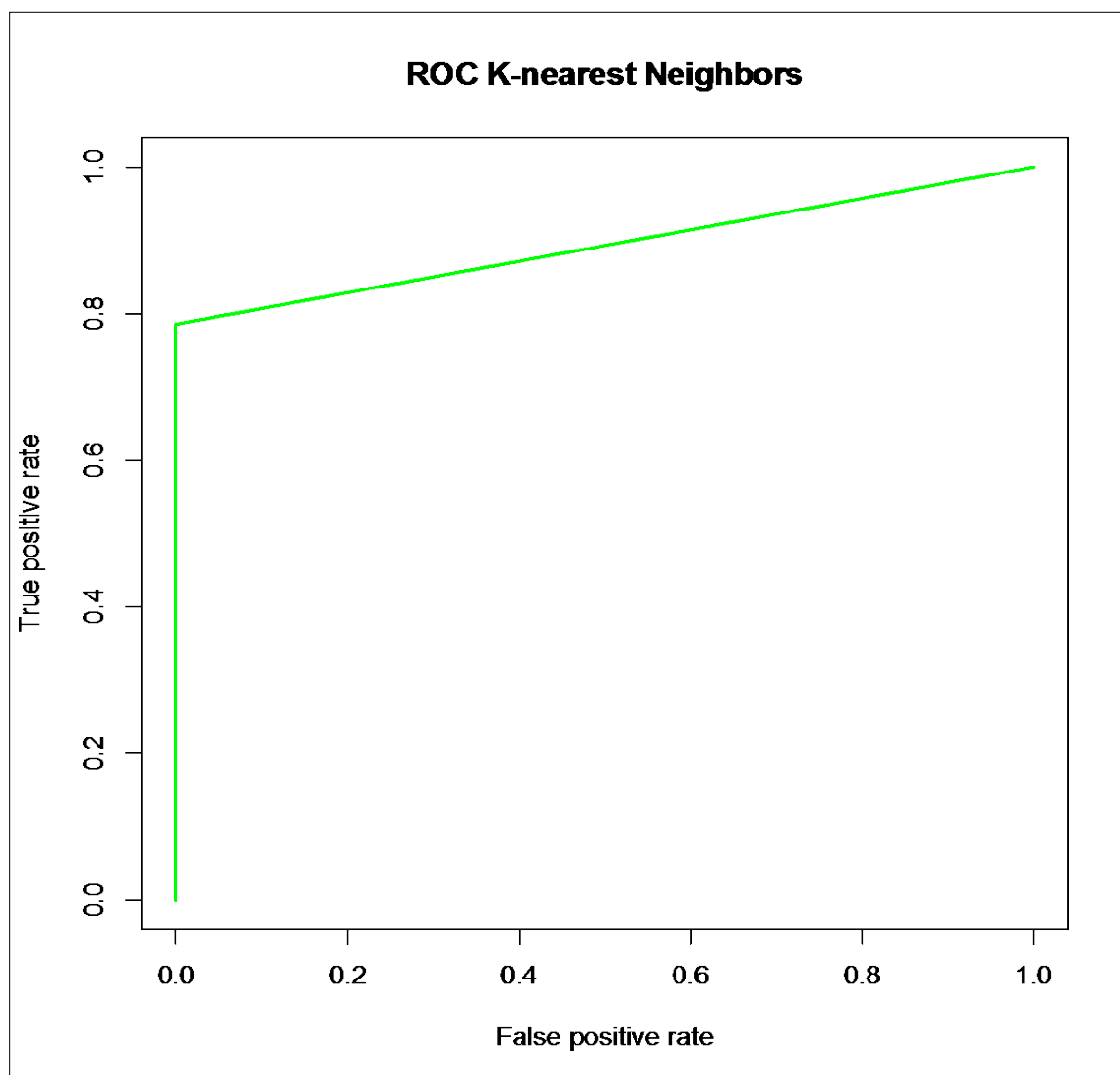


Figure 4.7: K-Nearest Neighbor: Leukemia dataset

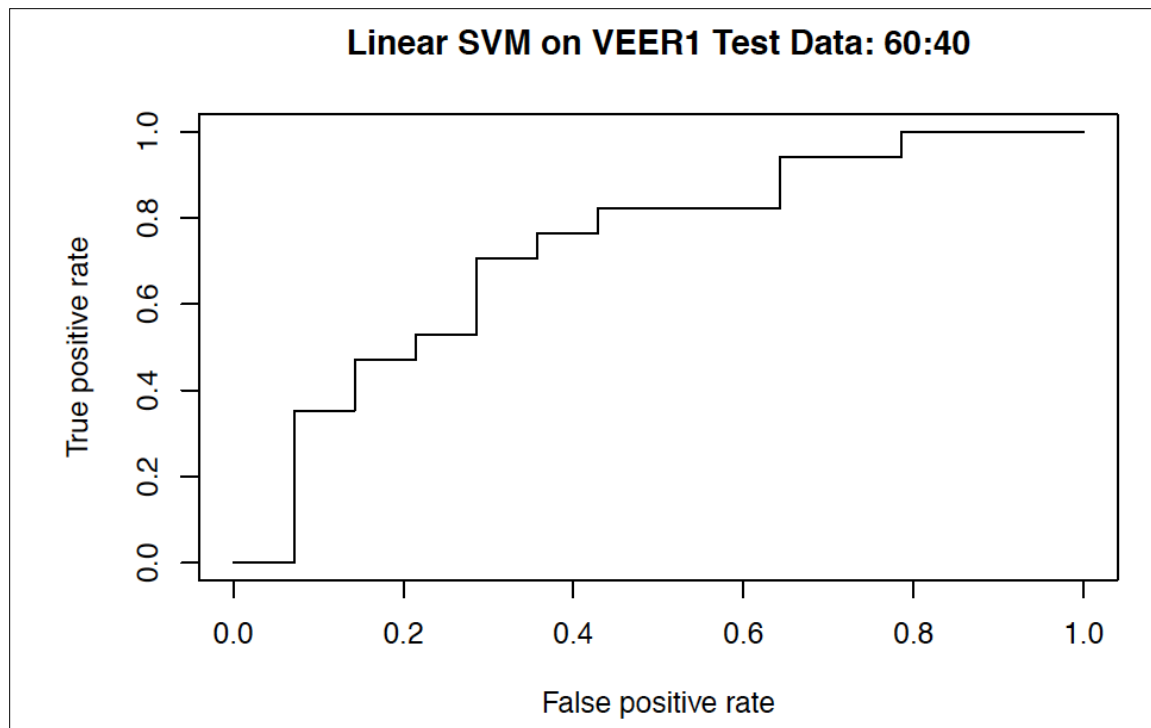
We calculated the area under curve for this classifier, which was 93%. The accuracy for this classifier with the leukemia dataset was 0.91.

### **4.3 Breast cancer dataset**

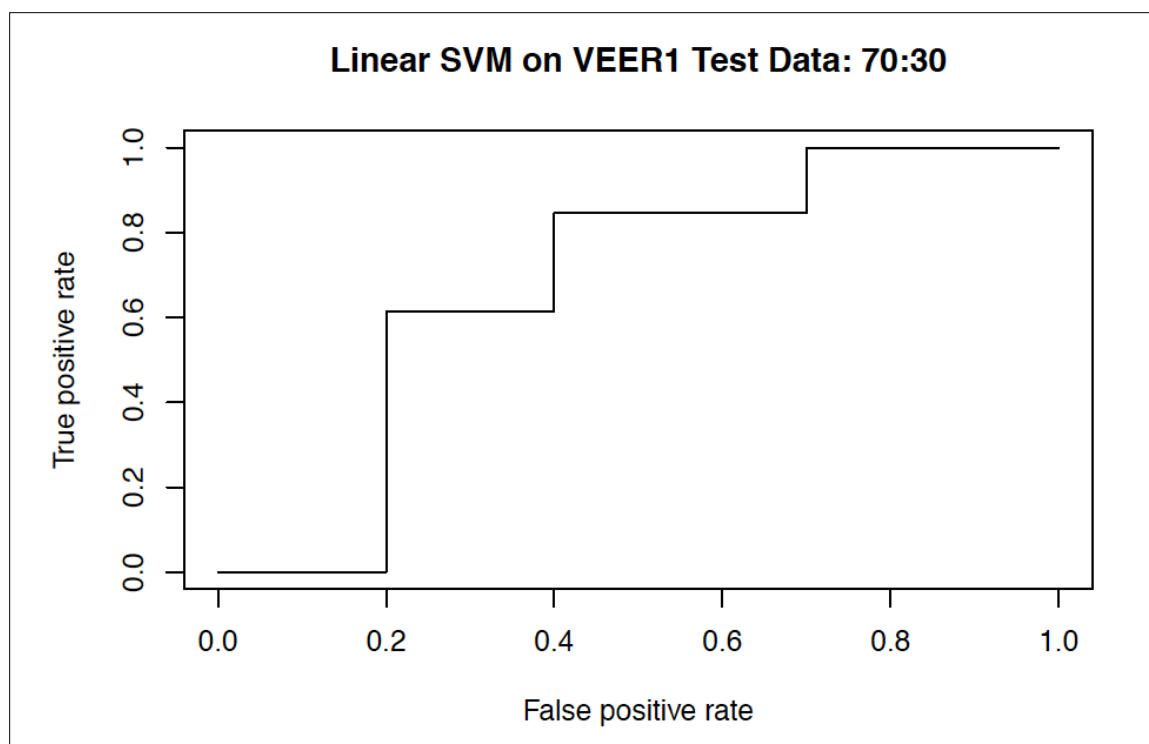
We have used a breast cancer dataset to examine the performance of three classification methods. With the breast cancer dataset we tried to determine the confidence level of classifying two groups of breast cancer patients; 34 samples from patients who developed distant metastases (DM) within 5 years and 44 samples from patients who continue to be disease-free (NODM) after a period of at least 5 years. We trained the three classifiers with this pre-classified data and the best classification method was determined.

#### **4.3.1 Training the SVM with Breast cancer dataset**

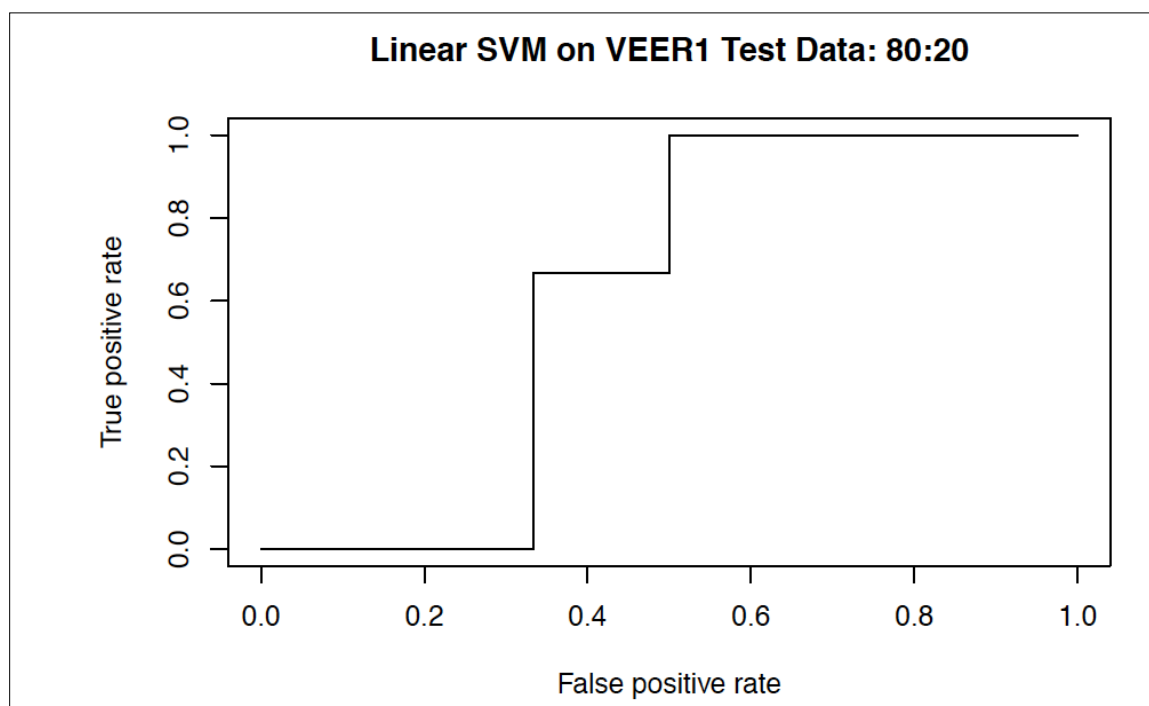
We trained linear kernel SVM to classify the breast cancer dataset. The cost of misclassification was chosen and tried with 10-fold cross validation to return the SVM with best parameters. Then the best model of training set was used to predict the class of samples in the test set. We repeated the process four times with different splits for the dataset to train and test the dataset. ROC curve and area under curve were calculated to compare the classifier performance with different split for the dataset. Figures 4.8.a, 4.8.b, 4.8.c, 4.8.d display ROC curve for SVM classifier for each distribution of the dataset to train dataset and test dataset.



**Figure 4.8.a: Linear SVM: breast dataset (60:40)**

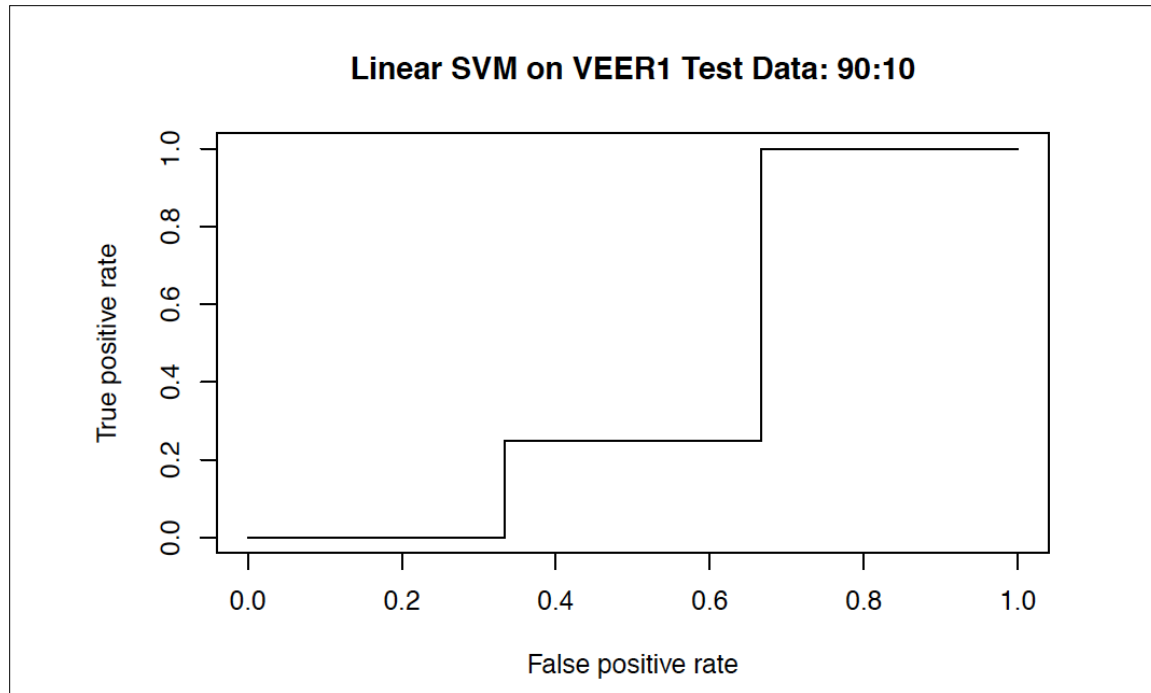


**Figure 4.8.b: Linear SVM: breast dataset (70:30)**



**Figure 4.8.c: Linear SVM: breast dataset (80:20)**





**Figure 4.8.d: Linear SVM: breast dataset (90:10)**

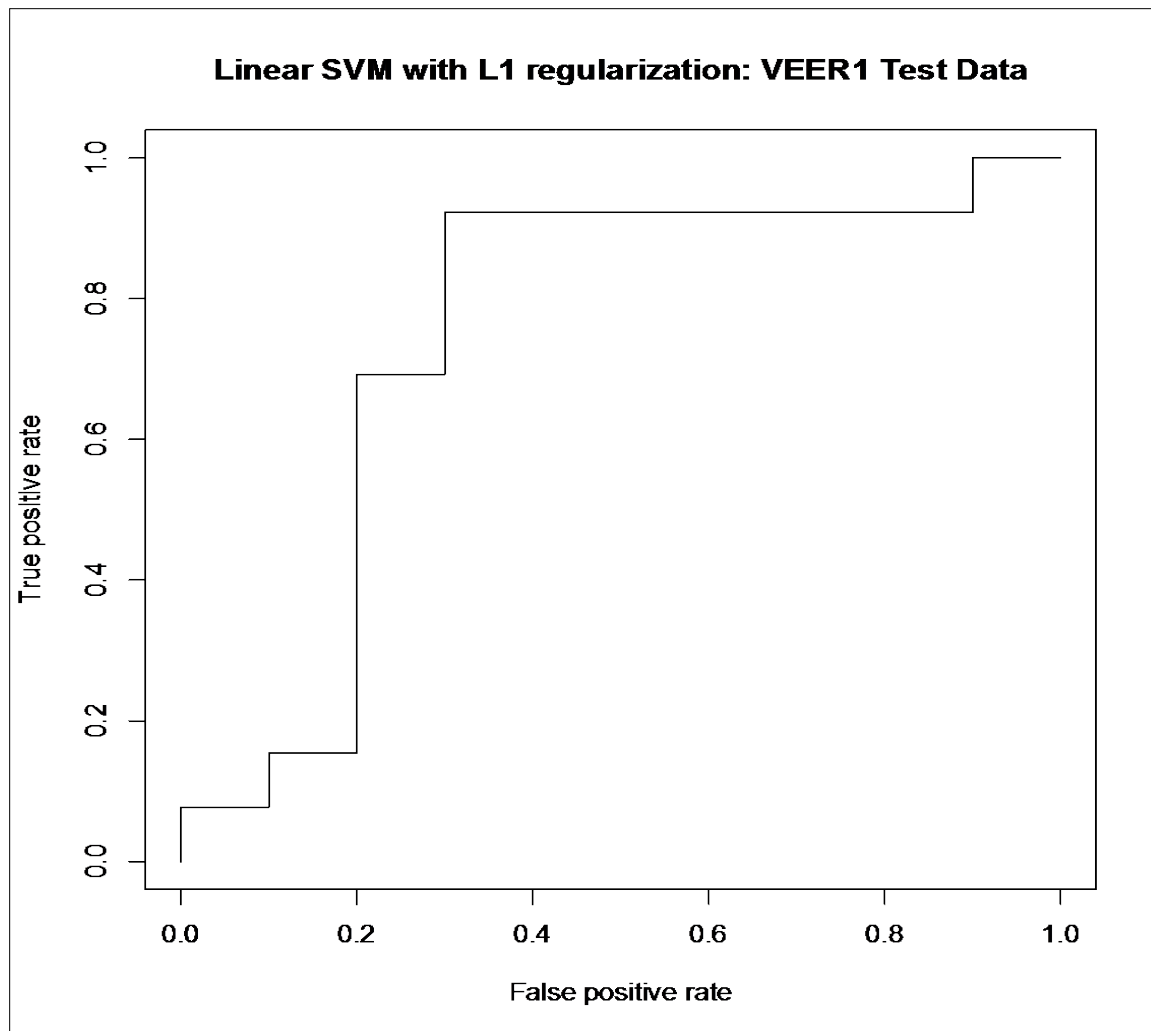
For each ROC curve we calculated the area under the curve. As shown in Fig. 4.8.a, the area under the curve for the distribution of 60% for training and 40% for test dataset was 0.72. Fig. 4.1.b displays the ROC curve for the classifier with 70% training and 30% testing and the AUC was 0.67. Fig. 4.1.c displays the ROC curve for the classifier with 80% training and 20% testing and the AUC was 0.61. Finally Fig. 4.1.d displays the ROC curve for the classifier with 90% training and 10% testing and the AUC was 0.41. When we compared the AUC between the groups of data, the distribution of 60% training and 40% testing gave the highest AUC, which was 72%. For this result the split of (60:40) was chosen to complete the process. After that we used the confusion matrix to evaluate performance of the classifier. The confusion matrix shows that six samples out of ten from the DM class were misclassified. For the NODM class two samples out of thirteen were misclassified.

		Predicted class		
Actual Class		DM	NODM	total
	DM	4	2	6
	NODM	6	11	17
	total	10	13	23

The accuracy for this classifier with breast cancer dataset was 0.65.

#### 4.3.2 SVM with L1, L2- Regularization for breast cancer dataset

We used L1, L2 regularization with SVM to improve the classifier performance with the breast dataset. We ran the model with 10-fold cross validation, and then we re-trained the best model with best-cost value. The best model was used to predict the class of samples in the test set. Figures 4.9.a and 4.9.b display ROC curve for SVM with L1 and L2 regularization.

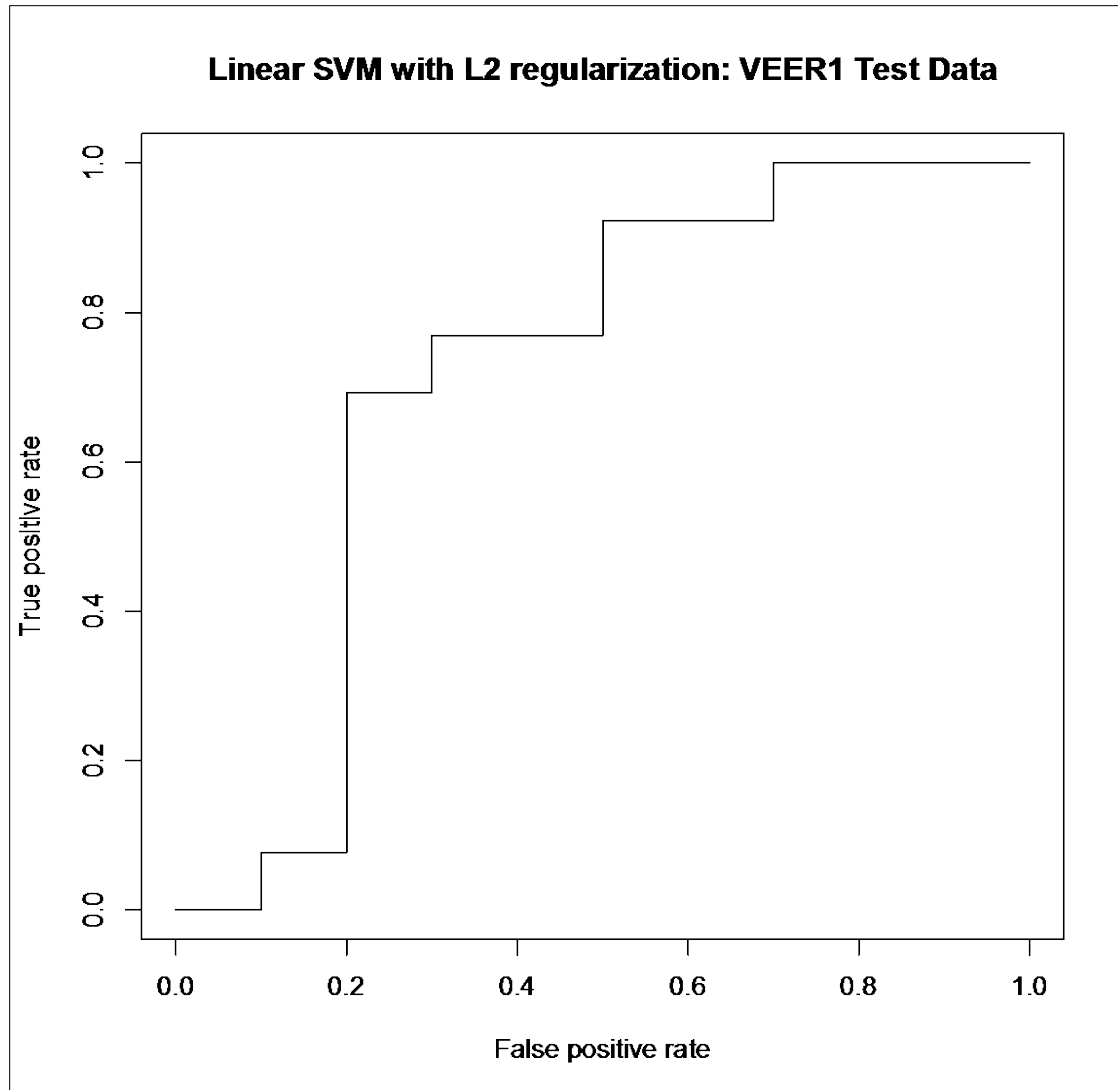


**Figure 4.9.a: Linear SVM with L1 regularization: breast dataset**

The confusion matrix for SVM with L1 regularization showed seven misclassified samples; six were from the DM class and one came from the NODM class.

		Predicted class		
		DM	NODM	total
Actual Class	DM	4	1	5
	NODM	6	12	17
	total	10	13	23

The accuracy of the classifier was 69%, with AUC 74%.



**Figure 4.9.b: Linear SVM with L2 regularization: breast dataset**

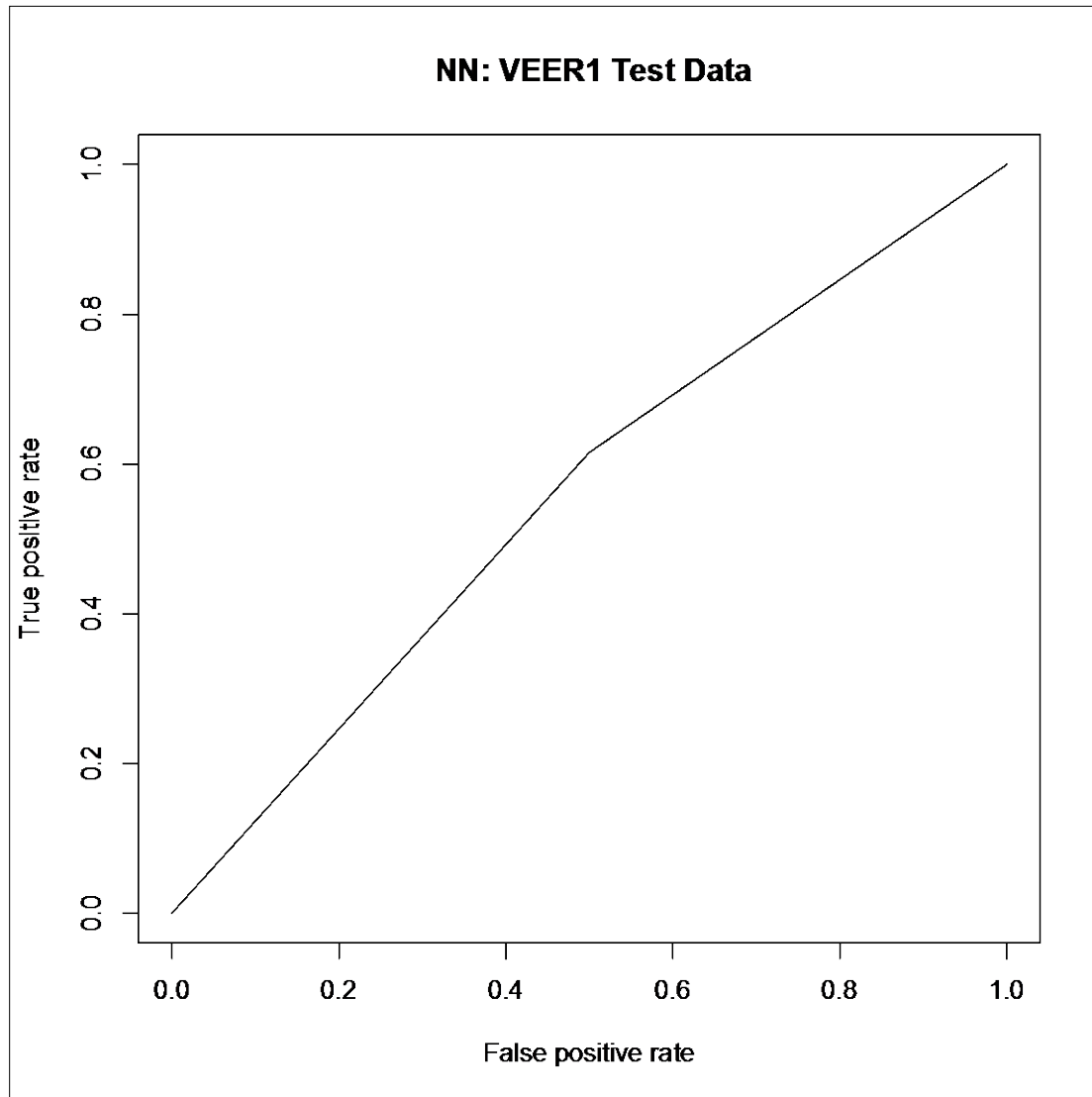
The confusion matrix for SVM with L2 regularization showed seven misclassified samples; five were from DM class and one came from NODM class.

		Predicted class		
		DM	NODM	total
Actual Class	DM	5	1	5
	NODM	5	12	17
total		10	13	23

The accuracy of the classifier is 73%, with AUC 72%.

### 4.3.3 Training Neural Network for breast cancer dataset

We trained the neural network classifier with the breast cancer dataset. After doing the pre-processing steps, the distribution of the data for 60% training and 40% testing was used. We normalized the values to be predicted and then trained a NN for the training data. ROC curve shown in Figure 4.10 was used to evaluate the performance of NN for the breast cancer dataset.



**Figure 4.10: Neural Network: breast dataset**

We calculated the area under curve (AUC) to evaluate the classifier, which was 55%. The confusion matrix showed that five samples out of ten from the DM class were misclassified. Five samples out of thirteen from NODM class were misclassified.

		Predicted class		
Actual Class		DM	NODM	total
	DM	5	5	10
	NODM	5	8	13
	total	10	13	23

The accuracy for this classifier with colon dataset was 0.56.

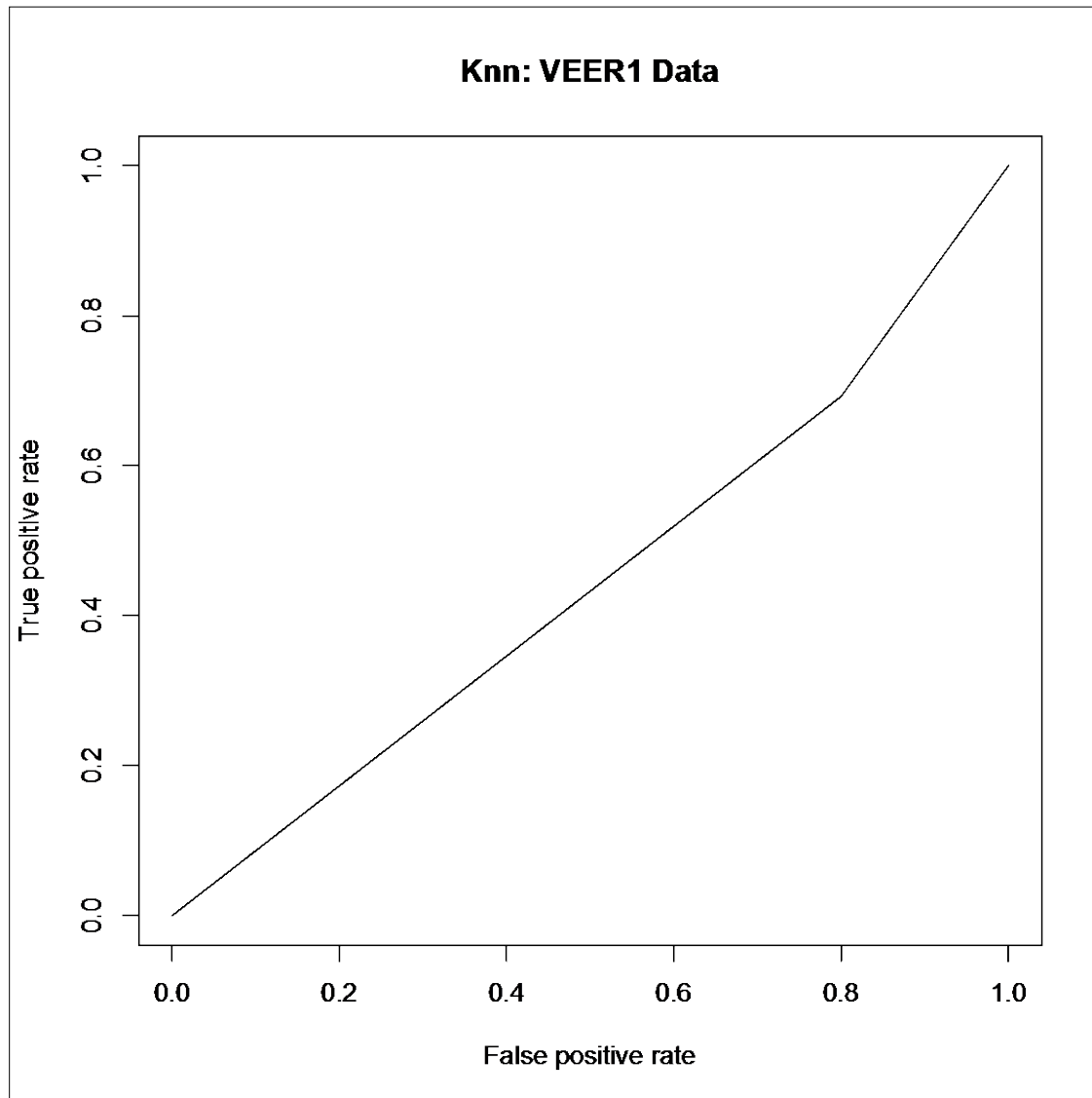
#### 4.3.4 Training K-Nearest Neighbor for breast cancer dataset

We trained K-Nearest Neighbor classifier for breast cancer dataset. Same pre-processed data were used and samples were split into 40% for test dataset and 60% for training dataset. We set different k values to choose the best number of neighbors for KNN, which gives the smallest test error rates. We calculated the confusion matrix to evaluate performance of the classifier with each k value.

With k=1 the confusion matrix shows that the false positive rate for DM class was nine out of ten, and the false positive rate for NODM class was five out of thirteen. For k=2 the false positive rate for DM class was eight out of ten, and the false positive rate for NODM class was five out of thirteen. The false positive rate for DM class with k=3 was eight out of ten and four out of thirteen for the NODM class. Finally, with k=4 the false positive rate was the same as k=3. Based on these outputs, k=3 yields the smallest test error rates. Therefore, we chose k=3 as the best number of neighbors for KNN with this breast cancer dataset.

		Predicted class		
Actual Class		DM	NODM	total
	DM	2	4	6
	NODM	8	10	18
	total	10	14	24

Classification result based on k=3 is shown in the ROC curve plot of Figure 4.11.



**Figure 4.11: K-Nearest Neighbor: breast dataset**

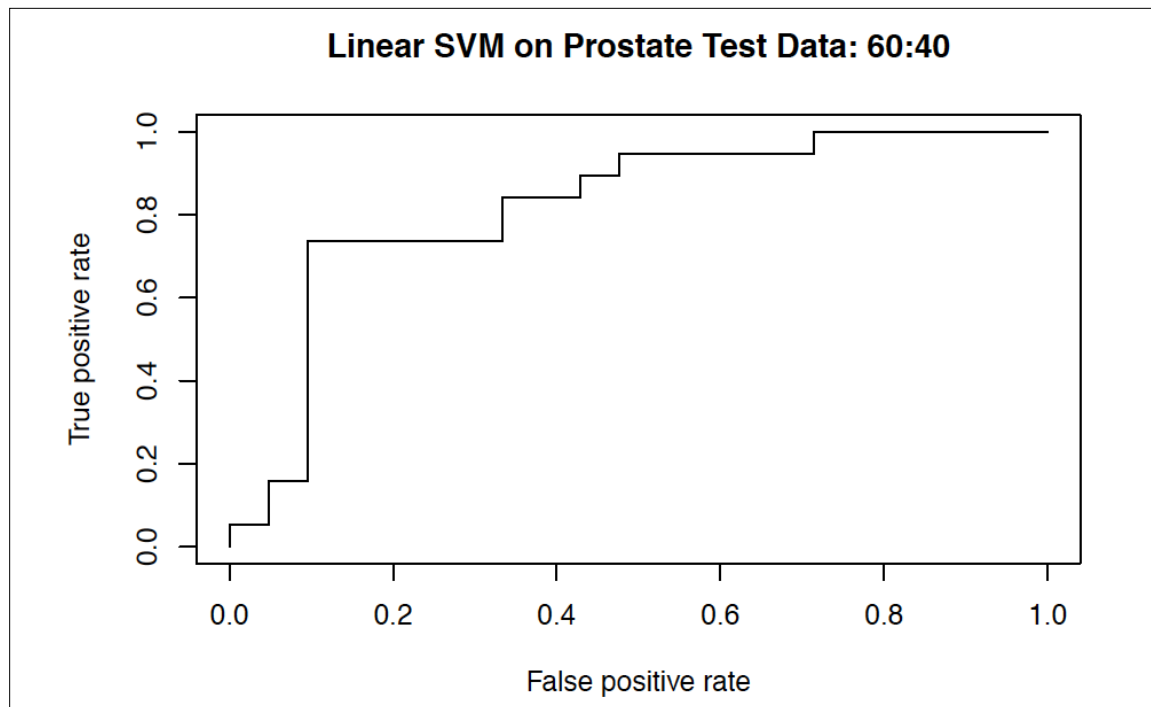
We calculated the area under curve (AUC) for this classifier, which was 43%. The accuracy for this classifier with the breast cancer dataset was 0.47.

## 4.4 Prostate cancer dataset

We have also used a prostate cancer dataset to examine the performance of the three classification methods. With the prostate cancer dataset we tried to determine the confidence level of classifying two groups of prostate cancer patients and healthy men. There were 52 prostate cancer patients and 50 healthy men. We trained the three classifiers with this pre-classified data and the best classification method was determined.

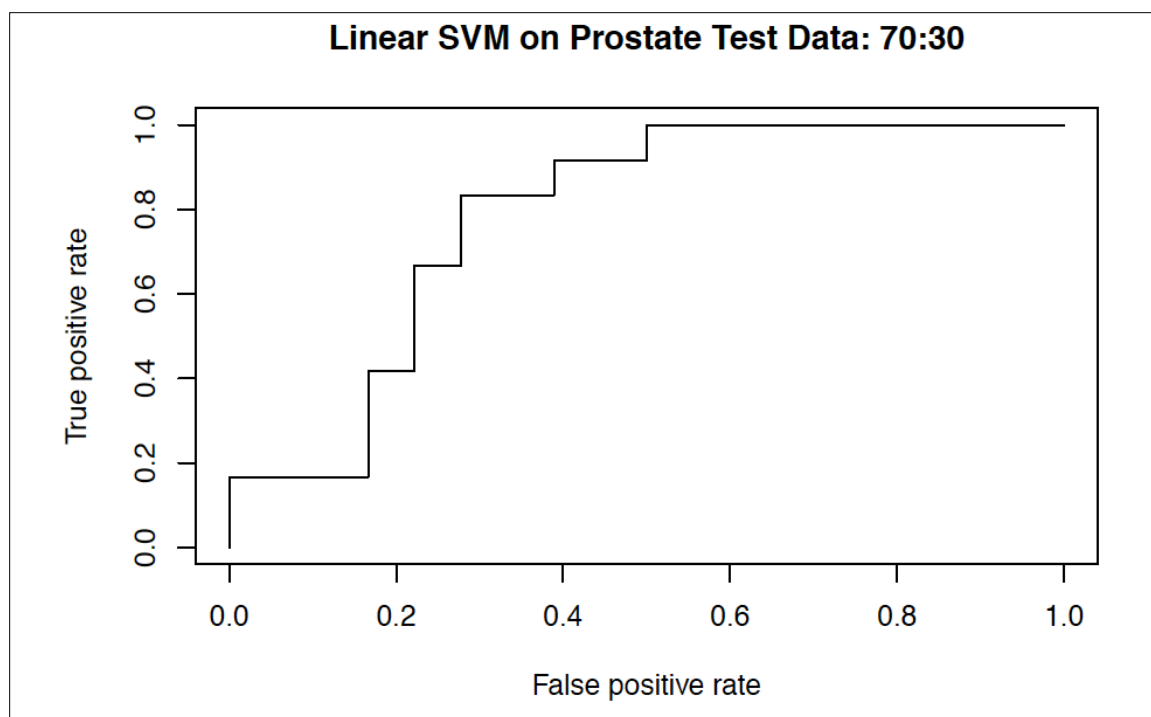
### 4.4.1 Training the SVM with Prostate cancer dataset

We trained linear kernel SVM to classify prostate cancer datasets. The cost of misclassification was chosen and tried with 10-fold cross validation to return the SVM with best parameters. Then the best model of training set was used to predict the class of samples in the test set. We repeated the process four times with different splits for the dataset to train and test dataset. ROC curve and area under curve were calculated to compare the classifier performance with different splits for the dataset. Figures 4.12.a, through 4.12.d display ROC curve for SVM classifier for each distribution of the dataset to train and test dataset.

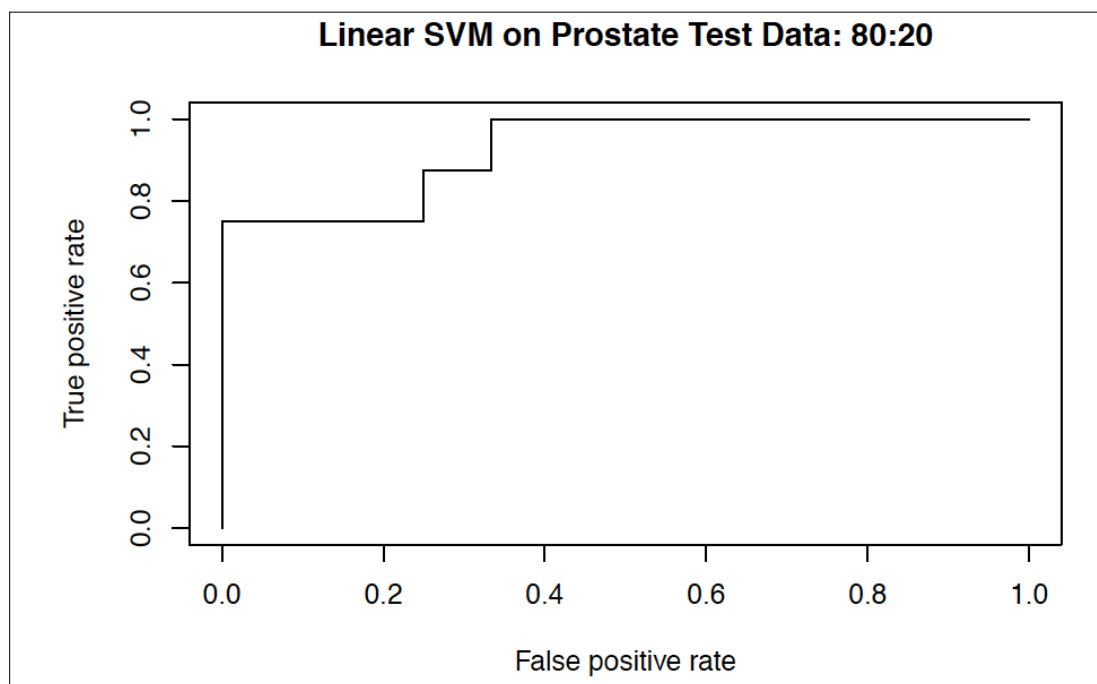


**Figure 4.12.a: Linear SVM: prostate dataset (60:40)**

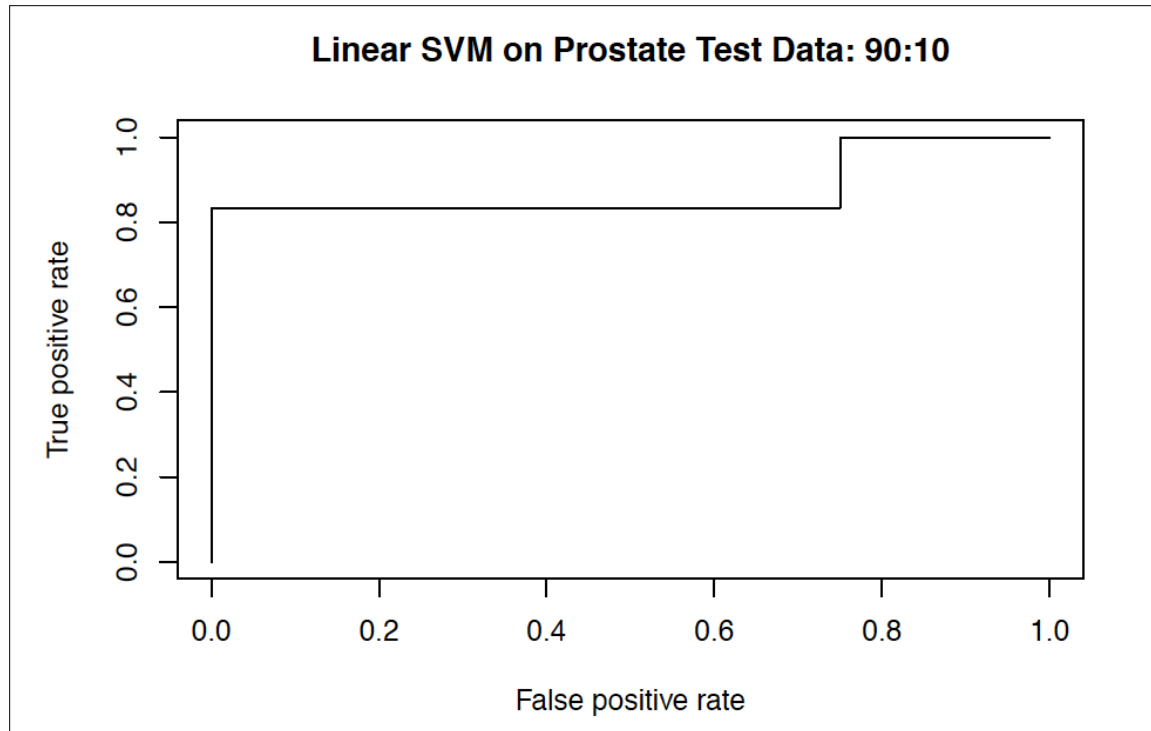




**Figure 4.12.b: Linear SVM: prostate dataset (70:30)**



**Figure 4.12.c: Linear SVM: prostate dataset (80:20)**



**Figure 4.12.d: Linear SVM: prostate dataset (90:10)**

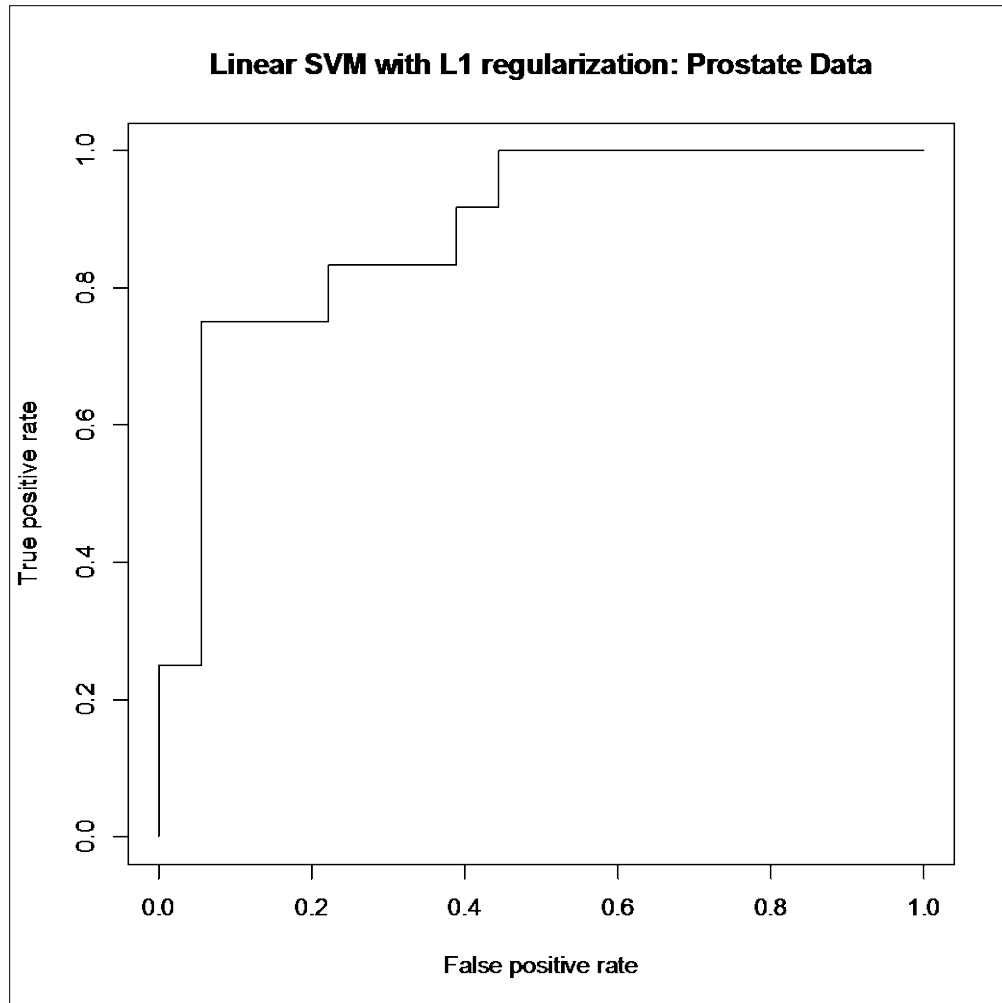
For each ROC curve we calculated the area under the curve. As shown in Fig. 4.12.a, the area under the curve (AUC) for the distribution of 60% for training and 40% for test dataset was 0.70. Fig. 4.12.b displays the ROC curve for the classifier with 70% training and 30% testing and the AUC was 0.78. Fig. 4.12.c displays the ROC curve for the classifier with 80% training and 20% testing and the AUC was 0.76. Finally, Fig. 4.12.d displays the ROC curve for the classifier with 90% training and 10% testing and the AUC was 0.75. When we compared the AUC between the groups of data, the distribution of 70% training and 30% testing gave the highest AUC, which was 78%. For this result the split of (70:30) was chosen to complete the process. After that, we used the confusion matrix to evaluate performance of the classifier. The confusion matrix shows that seven samples out of eighteen from the cancer class were misclassified. For the healthy class two samples out of twelve were misclassified.

		Predicted class		
Actual Class		Cancer	healthy	total
	Cancer	11	2	13
	healthy	7	10	17
	total	18	12	30

The accuracy for this classifier with the prostate dataset was 0.76.

#### 4.4.2 SVM with L1, L2- Regularization for prostate cancer dataset

When we trained SVM classifier with prostate cancer data the accuracy of the classifier was 76% with AUC 78%. Then we used L1, L2 regularization with SVM to improve the classifier's performance. We ran the model with 10-fold cross validation, then we re-trained best model with best-cost value. The best model was used to predict the class of samples in the test set. Figures 4.13.a and 4.13.b displays ROC curve for SVM with L1 and L2 regularization.

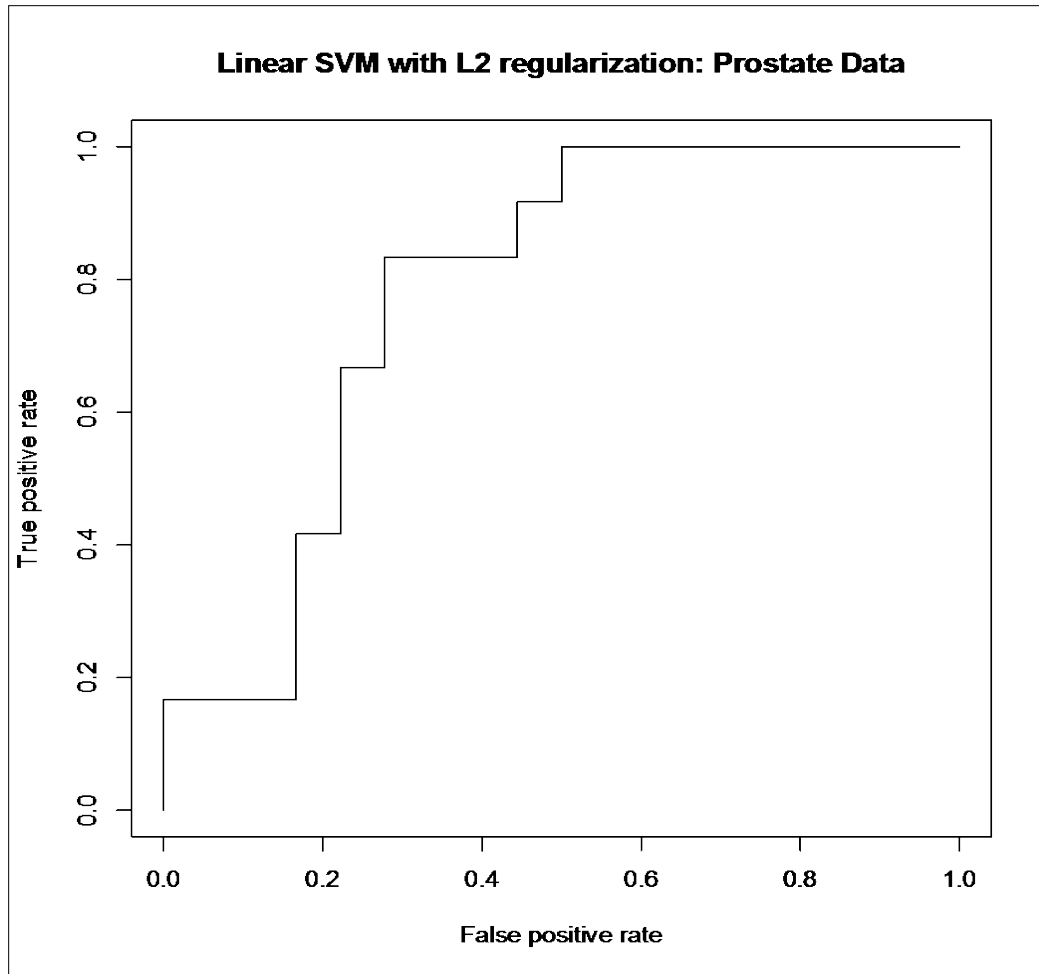


**Figure 4.13.a: Linear SVM with L1 regularization: prostate dataset**

The confusion matrix for SVM with L1 regularization showed six misclassified samples, four from the cancer class and two from the healthy class.

		Predicted class		
		Cancer	healthy	total
Actual Class	Cancer	14	2	16
	healthy	4	10	14
	total	18	12	30

The accuracy of the classifier was 80%, with AUC 88%.



**Figure 4.13.b: Linear SVM with L2 regularization: prostate dataset**

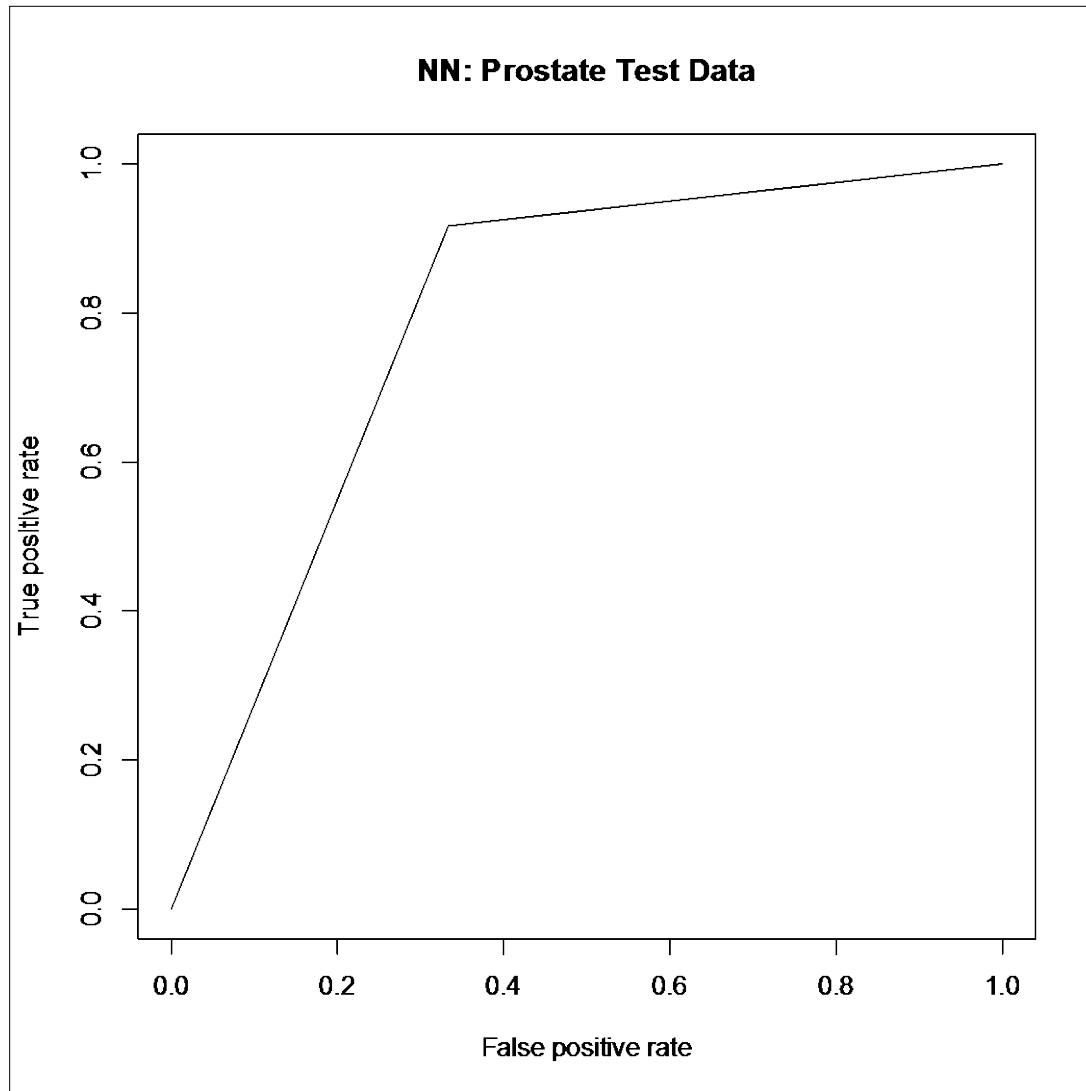
The confusion matrix for SVM with L2 regularization showed seven misclassified samples; five were from the cancer class and two came from the healthy class.

		Predicted class		
		Cancer	healthy	total
Actual Class	Cancer	13	2	13
	healthy	5	10	17
	total	18	12	30

The accuracy of the classifier was 76%, with AUC 77%

#### 4.4.3 Training Neural Network for prostate cancer dataset

We trained neural network classifier with prostate dataset. After doing the pre-processing step, the distribution of the data came to 70% training and 30% testing. We normalized the value to be predicted and trained the NN for the training data. ROC curve displayed in Figure 4.14 was used to evaluate the performance of NN for prostate cancer dataset.



**Figure 4.14: Neural Network: prostate dataset**

We calculated the area under the curve (AUC) to evaluate the classifier, which was 77%. The confusion matrix shows that six samples out of eighteen from the cancer class were misclassified. One sample out of twelve from healthy class was misclassified.

		Predicted class		
		Cancer	healthy	total
Actual Class	Cancer	12	1	13
	healthy	6	11	17
	total	18	14	30

The accuracy for this classifier with prostate dataset was 0.75.

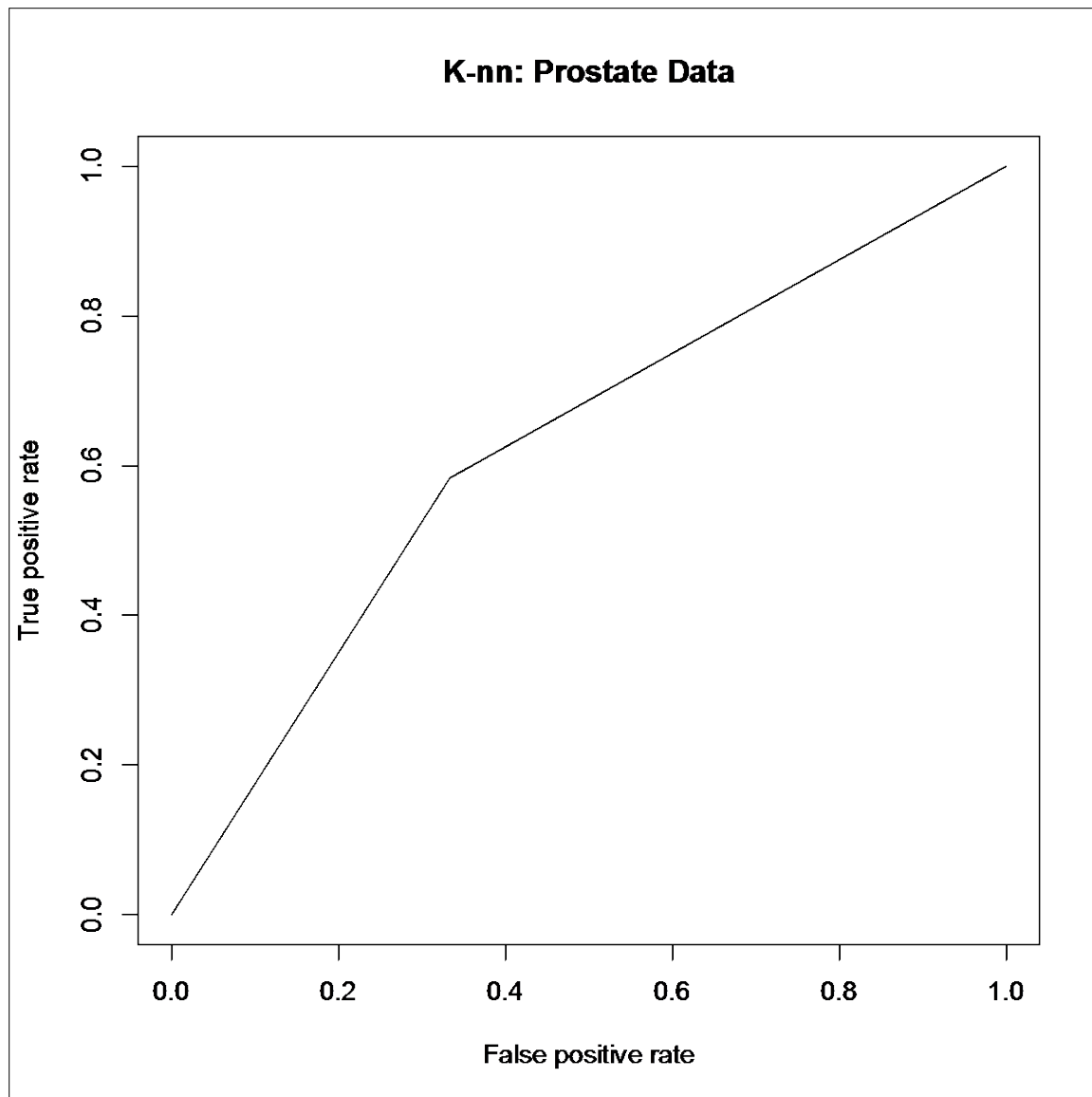
#### 4.3.4 Training K-Nearest Neighbor for prostate cancer dataset

We trained K-Nearest Neighbor classifier with prostate dataset. Same pre-processed data were used and samples were split into 30% for test dataset and 70% for training dataset. We set different k values to choose the best number of neighbors for KNN, which gives the smallest test error rates. We calculated the confusion matrix to evaluate performance of the classifier with each k.

With k=1 the confusion matrix shows that the false positive rate for the cancer class was six out of eighteen, and the false positive rate for the healthy class was five out of twelve. For k=2 the false positive rate for cancer class was six out of eighteen, and the false positive rate for the healthy class was seven out of twelve. The false positive rate for the cancer class with k=3 was three out of eighteen and eight out of twelve for the healthy class. Finally, with k=4 the false positive rate was three out of eighteen for the cancer class, and seven out of twelve for the healthy class. Based on these outputs, k = 4 yielded the smallest test error rates. Therefore, we chose k = 4 as the best number of neighbors for KNN with this prostate cancer dataset.

		Predicted class		
		Cancer	healthy	total
Actual Class	Cancer	10	3	13
	healthy	7	10	17
	total	17	13	30

The classification result based on k=4 is shown in the ROC curve plot of Figure 4.15.



**Figure 4.15: K-Nearest Neighbor: prostate dataset**

We calculated the area under curve for this classifier, which was 65%. The accuracy for this classifier with the prostate cancer dataset was 0.66.



## 4.2 Discussion

The results presented in this chapter clearly suggest that the SVM gives accurate results with high accuracies and the number of misclassifications drastically lowers when SVM classification was performed with the L1 and L2 regularization.

### 4.2.1 Evaluating the classifiers Performance

The classifiers were evaluated by comparing the accuracies of each classifier. Table 4.1 compares the confusion matrices (error rates) of each classifier and the area under curve for the colon dataset.

**Table4.1: Comparison of classification accuracies for colon dataset**

Methods	Confusion matrix				Accuracy	Area under curve
	FP	FN	TP	TN		
SVM	1	2	5	10	83	90
SVM-L1	2	1	6	9	83	93
SVM-L2	2	1	6	9	83	93
NN	2	2	5	9	77	76
KNN	1	4	3	10	72	73

The methods are linear kernel basis function SVM, neural network and k nearest neighbor. The six columns are the false positive, false negative, true positive, true negative, followed by accuracy and area under curve.

Table 4.2 compares the confusion matrices (error rates) of each classifier and the area under curve for the leukemia dataset.

**Table4.2: Comparison of classification accuracies for leukemia dataset**

Methods	Confusion matrix				Accuracy	Area under curve
	FP	FN	TP	TN		
SVM	1	0	20	13	97	100
NN	4	1	19	10	85	86
KNN	3	0	20	11	91	93

Similarly, table 4.3 compares the confusion matrices (error rates) of each classifier and the area under curve for the breast dataset.

**Table4.3: Comparison of classification accuracies for breast dataset**

Methods	Confusion matrix				Accuracy	Area under curve
	FP	FN	TP	TN		
SVM	2	6	4	11	65	72
SVM-L1	1	6	4	12	69	74
SVM-L2	1	5	5	12	73	74
NN	5	5	5	8	56	55
KNN	4	8	2	10	47	43

Table 4.4 compares the confusion matrices (error rates) of each classifier and the area under curve for the prostate dataset.

**Table4.4: Comparison of classification accuracies for prostate dataset**

Methods	Confusion matrix				Accuracy	Area under curve
	FP	FN	TP	TN		
SVM	2	7	11	10	76	78
SVM-L1	2	4	14	10	80	88
SVM-L2	2	5	13	10	76	77
NN	1	6	12	11	75	77
KNN	7	3	15	5	66	65

Tables 4.1, 4.2, 4.3 and 4.4 suggest that in most of the cases, the SVM performs better than NN and KNN and the uses of L1 and L2 with SVM did improve SVM performance. Through in some datasets we had accuracy of SVM classifier equivalent to SVM-L1 and SVM-L2, but this improved in area under curve with SVM-L1 and SVM-L2 (refer to table 4.1). Also, in some datasets we had accuracy of SVM classifier better than SVM-L2 (refer to table 4.4). Some datasets had a high accuracy of SVM classifier, so there was no need to use regularization technique. Confusion matrix showed the error rates of the classification, and in all cases SVM had a small number (false positive rate) of misclassified samples. Table 4.5 shows the accuracies of the classifiers for all the datasets obtained from the Table 4.1 through 4.4. It clearly suggests that the SVM outperforms all the classifiers consistently.

**Table4.5: Comparison of classification accuracies for all dataset**

Dataset	SVM			NN	KNN
	SVM	SVM-L1	SVM-L2		
Colon	83	83	83	77	72
Leukemia	97	-	-	85	91
Breast	65	69	73	56	47
Prostate	76	80	76	75	66

For every dataset the best performing method is a support vector machine using the linear kernel basis. In three separate tests, the linear basis SVM performs better than neural network and k-nearest neighbor.

## **Chapter 5**

### **Conclusions**

The aim of the present study was to determine the confidence level in classifying an unknown gene sample, based on the microarray data using SVM and two other classifiers, NN and KNN. The classifiers were evaluated and had their performances compared with each other. SVM had outperformed the other classifiers consistently.

Two types of kernel techniques were used and compared. Linear Kernel was chosen according to the test error rate. The analysis of linear Kernel SVM was performed using different percentages of distribution of the dataset for training and testing datasets. We received the best result for colon-cancer data, with distribution of 70% training and 30% testing. For leukemia dataset, changing the data distribution did not affect the result which we had chosen for the split of 70:30, to complete the process. The distribution of 60% training and 40% testing was chosen for breast-cancer dataset which gave best results. Finally, the distribution of 70% training and 30% testing was chosen to complete the process with prostate dataset.

The cost of misclassification was chosen and tried with the 10-fold cross validation (LOOCV method) to return the SVM with best parameters. Then, the best training model set was used to predict the class of samples in the test set.

The L1 L2 regularization techniques were processed to solve the over-fitting issues and feature selection in classification. A confusion matrix and ROC curve was used to evaluate the classifier's performance.

The neural network was the second classifier that was used in this study and was compared with SVM classifier. The same distribution of the datasets split for SVM was used with NN. We normalized the value to be predicted and then trained NN for the training data. Area under the curve and the confusion matrix was calculated to evaluate the classifier.

The third classification method used in this study was KNN. Same distribution of the datasets split with other classifiers was used. Different k values were set to choose the best number of

neighbors for KNN, which gives the smallest test error rates. Confusion matrix was calculated to evaluate the performance of the classifier, with each  $k$  and the chosen  $k$  with the best number of neighbors.

The classifiers were evaluated by comparing the accuracies of each classifier. Confusion matrix showed the error rates of the classification, and in all cases, SVM had a small number (false positive rate) of misclassified samples. The highest accuracy of colon data was 83% with SVM classifier, while the accuracy of NN with the same data was 77% and KNN was 72%. Leukemia data had the highest accuracy of 97% with SVM, 85% with NN, and 91% with KNN. For breast data, the highest accuracy was 73% with SVM-L2, while the accuracy was 56% with NN and 47% with KNN. Finally, the highest accuracy of prostate data was 80% with SVM-L1, while the accuracy was 75% with NN and 66% with KNN.

To conclude, the results in this study clearly suggested that SVM performs better than NN and KNN. SVM gives accurate results with high accuracies, and the number of misclassifications drastically lowers when SVM classification was performed with the L1 and L2 regularization.

## Future work

The results of the current study give us a clear picture of the stability of SVM classifier approach for the classification of a microarray dataset. SVM classifier approach can be applied to several other datasets for classification purposes and the stability of the method analyzed. Results attained in a classification problem are very much dependent on the preprocessing method used for the dataset. It might be possible to increase the stability of the classification model by trying different data preprocessing techniques. We can also apply other techniques to improve the accuracy of SVM like particle swarm optimization (PSO) and then do a comparative study with the SVM-L1, SVM-L2. Applying the above variations we can get more results and hence analyze the variations in the classification of samples. The work can be extended with the addition of new classifier methods such as Bayesian etc. for increasing the classification power on newly generated dataset.

## References

- [1] Jawdat, D.; "The Era of Bioinformatics"; Information and Communication Technologies, ICTTA '06; 2nd, vol.1, no., pp.1860-1865, (2006).
- [2] Jacques Cohen; "Bioinformatics-an introduction for computer scientists"; ACM Computing Surveys; Vol 36, Issue 2, pp 122-158; (2004).
- [3] Microarray Definition; <http://www.ncbi.nlm.nih.gov/About/primer/microarrays.html>, last access date – 06/10/2011.
- [4] Definition of microarray; from  
[MEDICINENET.COM.http://www.medterms.com/script/main/art.asp?articlekey=30712](http://www.medterms.com/script/main/art.asp?articlekey=30712)
- [5] SupratimChoudhuri; "Microarrays in Biology and Medicine"; Journal of Biochemical and Molecular Toxicology; Volume 18, Issue 4; pp 171-179, (2004).
- [6] Eva Paszek; "Affymetrix Chip-Basic Concepts", from  
<http://cnx.org/content/m12387/latest/oligo.gif>
- [7] Tarca, Adi L., Roberto Romero, and Sorin Draghici. "Analysis of Microarray Experiments of Gene Expression Profiling." *American journal of obstetrics and gynecology* 195.2 (2006): 373–388. *PMC*. Web. (2011).
- [8] Leming Shi; "DNA Microarray (Genome Chip)--- Monitoring the Genome on a Chip", (2002), <http://www.gene-chips.com/>
- [9] Campbell, A. M. "DNA microarray methodology-flash animation." (2001).
- [10] *What Is a Microarray* (n.d.): n. pag. *DNA Microarray*, (2007). Web.  
<http://classes.soe.ucsc.edu/bme215/Spring09/PPT/BME215-10-DNAMicroarray.pdf>
- [11] G. J. Babu & E. D. Feigelson; "Statistical Challenges in Modern Astronomy II", (New York: Springer) (1997), pp. 135-148; <http://sundog.stsci.edu/rick/SCMA/node2.html>



- [12] Zhong, Wenyan. "Feature selection for cancer classification using microarray gene expression data." (2014).
- [13] Sung-Huai Hsieh, Zhenyu Wang, Po-Hsun Cheng, I-Shun Lee, Sheau-Ling Hsieh, Feipei Lai; "Leukemia Cancer Classification based on Support Vector Machine"; 8th IEEE International Conference on Industrial Informatics; pp 819-824, (2010).
- [14] J. Phan, R. Moffitt, J. Dale, J. Petros, A. Young, M. Wang; "Improvement of SVM Algorithm for Microarray Analysis Using Intelligent Parameter Selection"; Engineering in Medicine and Biology Society, 2005, IEEE-EMBS 2005; 27th Annual International Conference of the IEEE; vol., no., pp.4838-4841, (2005).
- [15] Dr. Sayad, Saed "Artificial Neural Network." *Artificial Neural Network*. N.p., (2010). [http://chem-eng.utoronto.ca/~datamining/dmc/artificial\\_neural\\_network.htm](http://chem-eng.utoronto.ca/~datamining/dmc/artificial_neural_network.htm)
- [16] Mihir Sewak; "Application of Committee Neural Networks for Gene Expression Based Leukemia Classification", Master Thesis, University of Akron, (2008).
- [17] Manik Dhawan; "Application of committee K-NN classifier for gene expression profile classification"; Master Thesis, University of Akron; (2008).
- [18] S. Mukherjee P. Tamayo, D. Slonim, A. Verri, J.P. Mesirov, T. Poggio; "Support Vector Machine Classification of Microarray Data"; AI Memo 1677, CBCL Paper No 182, MIT; (1998).
- [19] Terrence S. Furey, Nello Cristianini, Nigel Duffy, David W. Bednarski, Michèle Schummer, David Haussler; "Support vector machine classification and validation of cancer tissue samples using microarray expression data"; *Bioinformatics*, Oxford Journals, Vol 16 no. 10, pp 906-914; (2000). <http://bioinformatics.oxfordjournals.org/content/16/10/906.full.pdf+html>
- [20] Huilin Xiong, Xue-wen Chen; "Kernel-based distance metric learning for microarray data"; *BMC Bioinformatics*, **7**:299 doi:10.1186/1471-2105-7-299; (2006).
- [21] Sorg, J., Miller, N., & Schneewind, O.; "Substrate recognition of type III secretion machines

testing the RNA signal hypothesis". *Cellular Microbiology*, 9:1217–1225, (2005).

[22] Broad Institute, <http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi>, last access date: 02/28/2011

[23] Princeton University, <http://microarray.princeton.edu/oncology/affydata/index.html>

[24] van 't Veer LJ et al.; "Gene expression profiling predicts clinical outcome of breast cancer", *Nature* 415:530-536, (2002).

[25] D. Singh et al.; "Gene expression correlates of clinical prostate cancer behavior". *Cancer Cell* 1:203—209, (2002).

[26] Luna De Ferrari; "Mining housekeeping genes with a Naive Bayes classifier"; School of Informatics University of Edinburgh; (2005).

[27] Adrash Jose, Dale Mugler, Zhong-HuiDuan; "A gene selection method for classifying cancer samples using 1D discrete wavelet transform"; *Int. J. Computational Biology and Drug Desing*, Vol. 2. No. 4, pp 398-411, (2009).

[28] K. Yang, Z. Cai, J. Li, G. Lin; "A stable gene selection in microarray data analysis"; *BMC Bioinformatics*; 7:228; (2006)

[29] A. A. Antipova, P. Tamayo, T. Golub; "A strategy for Oligonucleotide microarray probe reduction"; *Genome Biology*; 3:12; (2002)

[30] Xu, Ping, Guy N. Brock, and Rudolph S. Parrish. "Modified linear discriminant analysis approaches for classification of high-dimensional microarray data." *Computational Statistics & Data Analysis* 53.5 (2009): 1674-1687.

[31] Wang, Yu, et al. "Gene selection from microarray data for cancer classification—a machine learning approach." *Computational biology and chemistry* 29.1 (2005): 37-46.

[32] Mukherjee, Sayan. "Classifying microarray data using support vector machines." *A practical approach to microarray data analysis*. Springer US, (2003). 166-185.

- [33] Malepati, Rakesh Choudary. "Classification algorithms for genomic microarray." (2010).
- [34] Singh, Yashwant Prasad. "Adaboost and SVM based cybercrime detection and prevention model." *Artificial Intelligence Research* 1.2 (2012): p117.
- [35] KR, Seeja. "Microarray Data Classification Using Support Vector Machine." *International Journal of Biometrics and Bioinformatics (IJBB)* 5.1 (2011): 10.
- [36] Avison, Matthew. *Measuring gene expression*. Garland Science, 2005.
- [37] Madeswaran, TamilSelvi, and GM Kadhar Nawaz. "A Comparative Analysis of classification of Micro Array Gene Expression Data using Dimensionality Reduction Techniques." *IJCER* 1.4 (2012): 192-201.
- [38] Witten, Ian H., and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, (2005).
- [39] Fawcett, Tom. "An introduction to ROC analysis." *Pattern recognition letters* 27.8 (2006): 861-874.
- [40] Zhu, Ji, and Hui Zou. "Variable selection for the linear support vector machine." *Trends in Neural Computation*. Springer Berlin Heidelberg, (2007). 35-59.
- [41] Ross, Mary E., et al. "Classification of pediatric acute lymphoblastic leukemia by gene expression profiling." *Blood* 102.8 (2003): 2951-2959.
- [42] Sridhar, AadithyaVishnampettai. "A Hybrid Classifier Committee Approach for Microarray Sample Classification". Diss. The University of Akron, (2011).
- [43] Wang, Zhenyu. "Neuro-fuzzy modeling for microarray cancer gene expression data." *First year transfer report, University of Oxford* (2005).
- [44] Romualdi, Chiara, et al. "Pattern recognition in gene expression profiling using DNA array: a comparative study of different statistical methods applied to cancer classification." *Human Molecular Genetics* 12.8 (2003): 823-836.

- [45] Shieh, Grace S., Chy-Huei Bai, and Chih Lee. "Identify Breast Cancer Subtypes by Gene Expression Profiles." *Journal of Data Science* 2.2 (2004): 165-175.
- [46] Abdul, Ameer Basha Shaik. SVM Classification and Analysis of Margin Distance on Microarray Data. Diss. The University of Akron, 2011.
- [47] KNN classifier. <http://quipu-strands.blogspot.ca/> last access date – 16/09/2014.

## APPENDIX A

### GENERATING TRAINING AND TEST DATA RANDOMLY

**## Colon-cancer dataset with different split for the data (60:40; 70:30; 80:20; 90:10)**

```
rm(list=ls())
```

```
library(LiblineaR)
```

```
library(caret)
```

```
library(class)
```

```
library(e1071)
```

```
library(ROCR)
```

```
library(Biobase)
```

```
library(genefilter)
```

```
library(colonCA)
```

```
# set path and load helper functions
```

```
proj_path = file.path(Sys.getenv("HOME"), "Projects/71_makke/phase2")
```

```
source(file.path(proj_path, "helper.R"))
```

```
# load data
```

```
data(colonCA)
```

```
# split into training and testing subsets
```

```
train.pct = c(0.6, 0.7, 0.8, 0.9)
```

```
for (pct in train.pct) {
```

```
  # get training and testing set for each pct
```

```
  lst = mk_train_test_df(colonCA, pct.train=pct, yvar="class")
```

```

# fit linear kernel svm on training set with 10-fold CV

set.seed(100)

tune.out = tune(svm, y~, data=lst$df.train, kernel="linear",
decision.values=TRUE,

               ranges = list(cost=c(0.001, 0.01, 0.05, 0.1, 1, 5, 10, 100)))

# choose the best model from 10-fold CV

bestmod = tune.out$best.model

# use the best model to predict on test set

ypred = predict(bestmod, lst$df.test, decision.values=TRUE)

# plot ROC Curve and calculate AUC

fitted = attributes(ypred)$decision.values

auc = rocplot(fitted, lst$df.test$y,

              main=paste0("Linear SVM on ColonCA Test Data: ",
pct*100, ":", (1-pct)*100))

              cat("Area Under the Curve: ", auc)

}

## Leukemia dataset with different split for the data (60:40; 70:30; 80:20; 90:10)

rm(list=ls())

library(LiblineaR)

library(caret)

library(class)

library(e1071)

library(Biobase)

```

```

library(genefilter)

library(ROCR)

library(golubEsets)

if (!require(a4Base)) {

  source("http://bioconductor.org/biocLite.R")

  biocLite("a4Base")

} else {library(a4Base)}

# set path and load helper functions

proj_path = file.path(Sys.getenv("HOME"), "Projects/71_makke/phase2")

source(file.path(proj_path, "helper.R"))

# load data

data(Golub_Train)

data(Golub_Test)

Golub = combineTwoExpressionSet(Golub_Train, Golub_Test)

# dim(Golub) == dim(Golub_Train) + dim(Golub_Test)

# split into training and testing subsets

train.pct = c(0.6, 0.7, 0.8, 0.9)

for (pct in train.pct) {

  # get training and testing set for each pct

  lst = mk_train_test_df(Golub, pct.train=pct, yvar="ALL.AML")

  # fit linear kernel svm on training set with 10-fold CV

  set.seed(100)

  tune.out = tune(svm, y~., data=lst$df.train, kernel="linear",

```

```

decision.values=TRUE,

ranges = list(cost=c(0.001, 0.01, 0.05, 0.1, 1, 5, 10, 100)))

# choose the best model from 10-fold CV
bestmod = tune.out$best.model

# use the best model to predict on test set
ypred = predict(bestmod, lst$df.test, decision.values=TRUE)

# plot ROC Curve and calculate AUC

fitted = -attributes(ypred)$decision.values
auc = rocplot(fitted, lst$df.test$y,

main=paste0("Linear SVM on Golub Test Data: ",
pct*100, ":", (1-pct)*100))

cat("Area Under the Curve: ", auc)

}

## Breast-cancer dataset with different split for the data (60:40; 70:30; 80:20; 90:10)

rm(list=ls())

library(LiblineaR)

library(caret)

library(class)

library(e1071)

library(Biobase)

library(genefilter)

library(ROCR)

```



```

library(cancerdata)

# set path and load helper functions

proj_path = file.path(Sys.getenv("HOME"), "Projects/71_makke/phase2")

source(file.path(proj_path, "helper.R"))

# load data

data(VEER1)

# split into training and testing subsets

train.pct = c(0.6, 0.7, 0.8, 0.9)

for (pct in train.pct) {

  # get training and testing set for each pct

  lst = mk_train_test_veer1(VEER1, pct.train=pct, yvar="class")

  # fit linear kernel svm on training set with 10-fold CV

  set.seed(100)

  tune.out = tune(svm, y~., data=lst$df.train, kernel="linear",

  decision.values=TRUE,

  ranges=list(cost=c(0.00001, 0.001, 0.005, 0.01, 0.05,

  0.1, 0.2, 0.5, 1, 5, 10)))

  # choose the best model from 10-fold CV

  bestmod = tune.out$best.model

  # use the best model to predict on test set

  ypred = predict(bestmod, lst$df.test, decision.values=TRUE)

  # plot ROC Curve and calculate AUC

```

```

    fitted = attributes(ypred)$decision.values
auc = rocplot(fitted, lst$df.test$y,
              main=paste0("Linear SVM on VEER1 Test Data: ",
                           pct*100, ":", (1-pct)*100))
    cat("Area Under the Curve: ", auc)
}

## Prostate-cancer dataset with different split for the data (60:40; 70:30; 80:20; 90:10)

rm(list=ls())

library(LiblineaR)

library(caret)

library(class)

library(e1071)

library(genefilter)

library(ROCR)

library(Biobase)

library(sda)

# set path and load helper functions

proj_path = file.path(Sys.getenv("HOME"), "Projects/71_makke/phase2")

source(file.path(proj_path, "helper.R"))

# load data

data(singh2002)

X = singh2002$x

y = singh2002$y

```

```

train.pct = c(0.6, 0.7, 0.8, 0.9)

for (pct in train.pct) {

  # get training and testing set for each pct

  lst = mk_train_test_prostate(X, y, pct)

  # fit linear kernel svm on training set with 10-fold CV

  set.seed(100)

  tune.out = tune(svm, y~., data=lst$df.train, kernel="linear",
decision.values=TRUE,

                    ranges = list(c(0.00001, 0.001, 0.005, 0.01, 0.05, 0.1,
                                0.2, 0.5, 1, 5, 10)))

  # choose the best model from 10-fold CV

  bestmod = tune.out$best.model

  # use the best model to predict on test set

  ypred = predict(bestmod, lst$df.test, decision.values=TRUE)

  # plot ROC Curve and calculate AUC

  fitted = attributes(ypred)$decision.values

  auc = rocplot(fitted, lst$df.test$y,

                main=paste0("Linear SVM on Prostate Test Data: ",
pct*100, ":", (1-pct)*100))

  cat("Area Under the Curve: ", auc)

}

#####

```

```

## “ helper.R”

## BEGIN Defining Functions

mmfilt = function(r=5, d=500, na.rm=TRUE) {
  function(x) {
    minval = min(x, na.rm=na.rm)
    maxval = max(x, na.rm=na.rm)

    (maxval/minval>r) && (maxval-minval>d)

  }}

mmfun = mmfilt()

ffun = filterfun(mmfun)

get_sub = function(eset, f) {
  Wlow = 100
  Whigh = 16000

  x = exprs(eset)

  x[x<Wlow] = Wlow
  x[x>Whigh] = Whigh

  genefilter(x, f)

}

rocplot = function(pred, truth, ...) {
  predob = prediction(pred, truth)
  perf = performance(predob, "tpr", "fpr")

  plot(perf, ...)

# return AUC

```

```

auc.tmp = performance(predob, "auc")

auc = as.numeric(auc.tmp@y.values)

}

svm.regularization = function(X.train, y.train) {
  function(type, folds=10) {

    tryCosts = c(0.001, 0.01, 0.05, 0.1, 1, 5, 10, 100)

    bestCost = NA

    bestAcc = 0

    for (co in tryCosts) {

      acc = LiblineaR(data=X.train, labels=y.train, type=type,
        cost=co, bias=TRUE, cross=folds,
        verbose=FALSE)

      cat("Results for C=", co, ": ", acc, " accuracy.\n",
        sep="")

      if(acc>bestAcc){

        bestCost=co

        bestAcc=acc} }

      cat("Best cost is:", bestCost, "\n")

      cat("Best accuracy is:", bestAcc, "\n")

      c(bestCost=bestCost)

    } }

idx.train = function(df, pct.train=0.6) {

  n = ncol(df)

```

```

size_train = ceiling(n * pct.train)

set.seed(10384)

    sample(1:n, size_train)

}

mk_train_test_df = function(dat, pct.train=0.6, yvar) {

train = idx.train(dat, pct.train)

dat_train = dat[, train]

dat_test = dat[, -train]

# filter out genes

sub = get_sub(dat, ffun)

# subset the training and testing data using the filtered-out genes

dat_trainSub = dat_train[sub, ]

dat_testSub = dat_test[sub, ]

# transform training and test sets to data.frame

X.train = t(exprs(dat_trainSub))

X.test = t(exprs(dat_testSub))

# make training data frame

X.train = scale(X.train, center=TRUE, scale=TRUE) # scale X.train

y.train = factor(dat_trainSub[[yvar]])

df.train = data.frame(X.train, y=y.train)

# make testing data frame

X.test = scale(X.test, attr(X.train, "scaled:center"),

attr(X.train, "scaled:scale")) # scale X.test

```

```

y.test = factor(dat_testSub[[yvar]])

df.test = data.frame(X.test, y=y.test)

list(df.train=df.train, df.test=df.test)}

idx_col_w_NA= function(df) {

  idx = c()

  for (iin1:ncol(df))

    if (sum(is.na(df[, i])) >0) idx = c(idx, i)

  idx}

mk_train_test_prostate = function(X, y, pct.train=0.6) {

  # split into training and testing subsets

  n = length(y)

  size_train = ceiling(n * pct.train)

  set.seed(10384)

  train = sample(1:n, size_train)

  X.train = X[train, ]; y.train = y[train]

  X.test = X[-train, ]; y.test = y[-train]

  # make training data frame

  X.train = scale(X.train, center=TRUE, scale=TRUE) # scale X.train

  df.train = data.frame(X.train, y = y.train)

  # make testing data frame

  X.test = scale(X.test, attr(X.train, "scaled:center"),

    attr(X.train, "scaled:scale")) # scale X.test

  df.test = data.frame(X.test, y = y.test)

```

```

list(df.train=df.train, df.test=df.test)    }

mk_train_test_veer1 = function(dat, pct.train=0.6, yvar) {

  train = idx.train(dat, pct.train)

  dat_train = dat[, train]

  dat_test = dat[, -train]

  # transform training and test sets to data.frame

  X.train = t(exprs(dat_train))

  X.test = t(exprs(dat_test))

  delete.col.idx = c(idx_col_w_NA(X.train), idx_col_w_NA(X.test))

  # make training data frame

  X.train = X.train[, -delete.col.idx]

  X.train = scale(X.train, center=TRUE, scale=TRUE) # scale X.train

  y.train = factor(dat_train[[yvar]])

  df.train = data.frame(X.train, y=y.train)

  # make testing data frame

  X.test = X.test[, -delete.col.idx]

  X.test = scale(X.test, attr(X.train, "scaled:center"),
    attr(X.train, "scaled:scale")) # scale X.test

  y.test = factor(dat_test[[yvar]])

  df.test = data.frame(X.test, y=y.test)

  list(df.train=df.train, df.test=df.test)

}

## END Defining Functions

```



## APPENDIX B

### Preprocessing, transformations and filter the data

```
## BEGIN Defining Functions
```

```
mmfilt = function(r=5, d=500, na.rm=TRUE) {
```

```
function(x) { ## Filter methods select
```

```
## features according to
```

```
## criteria that are
```

```
## independent of those
```

```
## criteria that the classifier
```

```
## optimizes.
```

```
minval = min(x, na.rm=na.rm)
```

```
maxval = max(x, na.rm=na.rm)
```

```
(maxval/minval>r) && (maxval-minval>d)
```

```
}
```

```
}
```

```
mmfun = mmfilt()
```

```
ffun = filterfun(mmfun)
```

```
## Windsorizing the data (setting the minimum expression ## values to 100 and the maximum to 16000)
```

```
get_sub = function(eset, f) {
```

```
Wlow = 100
```

*Whigh = 16000*

*x = exprs(eset)*

*x[x < Wlow] = Wlow*

*x[x > Whigh] = Whigh*

*genefilter(x, f)}*

*# filter out genes (Leukemia dataset)*

*sub = get\_sub(Golub\_Train, ffun)*

*sum(sub)*

*# filter out genes (Colon cancer dataset)*

*sub = get\_sub(colonCA, ffun)*

*sum(sub)*

---

## APPENDIX C

### Training SVM classifier with Liner Kernel and Redial Kernel

#### *## radial kernel svm on training set with 10-fold CV*

```
set.seed(100)

tune.out = tune(svm, y~., data=df.train, kernel="radial",
               ranges=list(cost=c(0.001, 0.01, 0.05, 0.1, 1, 10, 100),
                           gamma=c(0.0001, 0.001, 0.01, 0.1,
                                   0.5, 1)))

summary(tune.out)
```

#### *## choose the best model from 10-fold CV*

```
bestmod = tune.out$best.model

summary(bestmod)
```

#### *## use the best model to predict on test set*

```
ypred = predict(bestmod, df.test)

table(predict=ypred, truth=df.test$y)
```

#### **# linear kernel svm on training set with 10-fold CV**

```
set.seed(100)

tune.out = tune(svm, y~., data=df.train, kernel="linear", decision.values=TRUE,
               ranges = list(cost=c(0.001, 0.01, 0.05, 0.1, 1, 5, 10, 100)))
```

```
summary(tune.out)

# choose the best model from 10-fold CV

bestmod = tune.out$best.model

summary(bestmod)

# use the best model to predict on test set

ypred = predict(bestmod, df.test, decision.values=TRUE)

( tbl = table(predict=ypred, truth=df.test$y) )

( test.error = (tbl[1,2] + tbl[2,1]) / sum(tbl) )
```

## APPENDIX D

### Display confusion Matrix

```
#input

library(caret)

lvs<- c("AML", "ALL")

truth<- factor(rep(lvs, times = c(14, 20)),

levels = rev(lvs))

pred<- factor(

  c(

    rep(lvs, times = c(13, 1)),

    rep(lvs, times = c(0, 20))),

levels = rev(lvs))

xtab<- table(pred, truth)

confusionMatrix(xtab)
```

---

## APPENDIX E

### Plot ROC Curve and calculate Area Under Curve

```
rocplot = function(pred, truth, ...) {  
  
  predob = prediction(pred, truth)  
  
  perf = performance(predob, "tpr", "fpr")  
  
    plot(perf, ...)  
  
  # return AUC  
  
  auc.tmp = performance(predob, "auc")  
  
  auc = as.numeric(auc.tmp@y.values)  
  
  # plot ROC Curve and calculate AUC  
  
  fitted = attributes(ypred)$decision.values  
  
  ( auc = rocplot(fitted, df.test$y, main="Linear SVM: ColonCA Test Data") )  
  
}
```

---

## APPENDIX F

### SVM with L1 regularization

**# svm with L1 regularization with 10-fold CV**

```
set.seed(100)
```

```
t = 3
```

```
svm.L1 = svm.regularization(X.train, y.train)
```

```
bestCost = svm.L1(type=t, folds=10)
```

```
# re-train best model with best cost value
```

```
bestmod = LiblineaR(data=X.train, labels=y.train, type=t, cost=bestCost,
```

```
bias=TRUE, verbose=FALSE)
```

```
# make prediction
```

```
ypred = predict(bestmod, X.test, proba=FALSE, decisionValues=TRUE)
```

```
# Display confusion matrix
```

```
( tbl = table(predict=ypred$predictions, truth=df.test$y) )
```

```
confusionMatrix(tbl)
```

```
fitted = ypred$decisionValues[,1]
```

```
( auc = rocplot(fitted, df.test$y,
```

```
main="Linear SVM with L1 regularization: ColonCA Test Data") )
```

### SVM with L2 regularization

```

# svm with L2 regularization with 10-fold CV

set.seed(100)

t = 1

svm.L2 = svm.regularization(X.train, y.train)

bestCost = svm.L2(type=t, folds=10)

# re-train best model with best cost value

bestmod = LiblinearR(data=X.train, labels=y.train, type=t, cost=bestCost,
bias=TRUE, verbose=FALSE)

# make prediction

ypred = predict(bestmod, X.test, proba=FALSE, decisionValues=TRUE)

# Display confusion matrix

( tbl = table(predict=ypred$predictions, truth=df.test$y) )

confusionMatrix(tbl)

fitted = ypred$decisionValues[,1]

( auc = rocplot(fitted, df.test$y,
main="Linear SVM with L2 regularization: ColonCA Test Data") )

```

---



## APPENDIX G

### Training Neural Network classifier

#### ## Training NN classifier with Colon-cancer dataset

```
> library(rpart)

> gdf<- data.frame(ALL = colonCA_trainSub$class, t(exprs(colonCA_trainSub)))

> tr1 <- rpart(ALL ~ ., data = gdf)

> print(summary(tr1))

set.seed(100)

library(nnet)

gdf2<- df.train[, c(1, sample(2:ncol(df.train), size = 150))]

nn1<- nnet(y~., data=df.train, size = 5, decay = 0.01, MaxNWts = 6131

## Predict on training set

> print(table(predict(nn1, type = "class"), y.train))

## Predict on test set

> print(table(predict(nn1, new = df.test,

+ type = "class"), y.test))

# confusionMatrix

lvs<- c("n","t")

truth <- factor(rep(lvs,times= c(11,7)),levels= rev(lvs))
```

```

pred<- factor(c(rep(lvs,times= c(9,2)),rep(lvs,times=
+ c(2,5))),levels= rev(lvs))

xtab<- table(pred, truth)

confusionMatrix(xtab)

# plot ROC Curve and calculate AUC

fitted = attributes(pred)$decision.values

( auc = rocplot(pred, truth, main="NN: ColonCA Test Data") )

## Training NN classifier with Breast-cancer dataset

library(rpart)

gdf<- data.frame(ALL = VEER1_train$class , t(exprs(VEER1_train)))

> tr1 <- rpart(ALL ~ ., data = gdf)

> print(summary(tr1))

> library(nnet)

> gdf2 <- gdf[, c(1, sample(2:ncol(gdf), size = 150))]

> nn1 <- nnet(ALL ~ ., data = gdf2, size = 5, decay = 0.01, MaxNWts = 5000)

## Predict on training set

> print(table(predict(nn1, new = data.frame(t(exprs(VEER1_train))),
+ type = "class"), VEER1_train$class))

## Predict on test set

(tbl = table(predict(nn1, new = data.frame(t(exprs(VEER1_test))),

```

```

+ type = "class"), VEER1_test$class))

# confusionMatrix

<library(caret)

<lvs<- c("n","t")

> truth <- factor(rep(lvs,times= c(13,27)),levels= rev(lvs))

>pred<- factor(c(rep(lvs,times= c(11,2)),rep(lvs,times= c(0,27))),levels= rev(lvs))

>xtab<- table(pred, truth)

>confusionMatrix(xtab)

# plot ROC Curve and calculate AUC

library(pROC)

plot.roc(ypred, truth,main=" NN: VEER1 Test Data")

## Training NN classifier with Prostate-cancer dataset

> library(rpart)

>gdf<- data.frame(ALL = singh2002$y, t(exprs(singh2002$ySub)))

> tr1 <- rpart(ALL ~ ., data = gdf)

> print(summary(tr1))

library(nnet)

> gdf2 <- gdf[, c(1, sample(2:ncol(gdf), size = 150))]

> nn1 <- nnet(ALL ~ ., data = gdf2, size = 5, decay = 0.01, MaxNWts = 5000)

```

```
## Predict on training set
```

```
> print(table(predict(nn1, type = "class"), df.train$y))
```

```
## Predict on test set
```

```
> print(table(predict(nn1, new = data.frame(X.train),
```

```
+ type = "class"), df.train$y))
```

```
# confusionMatrix
```

```
> ( tbl = table(predict(nn1, type = "class"), df.train$y) )
```

```
> confusionMatrix(tbl)
```

*Confusion Matrix and Statistics*

---

## Appendix H

### Training K-Nearest Neighbor classifier

#### ## Training KNN classifier with Colon-cancer dataset

```
> knn1 <- knn.cv(t(exprs(colonCA_trainSub)),colonCA_train$class, k = 1, prob = TRUE)
```

```
> print(table(knn1, colonCA_train$class))
```

Knn1	n	t
n	9	1
t	4	26

```
sum(knn1 == colonCA_train$class)/length(colonCA_train$class)
```

```
[1] 0.875
```

```
> knn1 <- knn.cv(t(exprs(colonCA_trainSub)),colonCA_train$class, k = 2, prob = TRUE)
```

```
> print(table(knn1, colonCA_train$class))
```

Knn1	n	t
n	8	1
t	5	26

```
> sum(knn1 == colonCA_train$class)/length(colonCA_train$class)
```

```
[1] 0.875
```

```
> knn1 <- knn.cv(t(exprs(colonCA_trainSub)),colonCA_train$class, k = 3, prob = TRUE)
```

```
> print(table(knn1, colonCA_train$class))
```

Knn1	n	t
n	9	1
t	4	26

```
> library(caret)
```

```
lvs<- c("n","t")
```

```
> truth <- factor(rep(lvs,times= c(13,27)),levels= rev(lvs))
```

```
>pred<- factor(c(rep(lvs,times= c(9,4)),rep(lvs,times= c(0,27))),levels= rev(lvs))
```

```
>xtab<- table(pred, truth)
```

```
>confusionMatrix(xtab)
```

### *Confusion Matrix and Statistics*

Knn1	n	t
n	27	4
t	0	9

*Accuracy : 0.9*

*95% CI : (0.7634, 0.9721)*

*No Information Rate : 0.675*

*P-Value [Acc> NIR] : 0.0009239*

*Kappa : 0.7523*

*Mcnemar's Test P-Value : 0.1336144*

*Sensitivity : 1.0000*

*Specificity : 0.6923*

*PosPred Value : 0.8710*

*NegPred Value : 1.0000*

*Prevalence : 0.6750*

*Detection Rate : 0.6750*

*Detection Prevalence : 0.7750*

*Balanced Accuracy : 0.8462*

*'Positive' Class : t*

**## Training KNN classifier with Leukemia cancer dataset**

```
> knn1 <- knn.cv(t(exprs(golubTrainSub)),gdf2$ALL, k = 1, prob = TRUE)
```

```
> print(table(knn1, Golub_Train$ALL))
```

Knn1	ALL	AML
ALL	27	3
AML	0	8

```
sum(knn1 == golubTrainSub$ALL)/length(golubTrainSub$ALL)
```

*[1] 0.921*

```
> knn1 <- knn.cv(t(exprs(golubTrainSub)),gdf2$ALL, k = 2, prob = TRUE)
```

```
> print(table(knn1, Golub_Train$ALL))
```

Knn1	ALL	AML
ALL	27	3
AML	0	8

```
sum(knn1 == golubTrainSub$ALL)/length(golubTrainSub$ALL)
```

```
[1] 0.921
```

```
> knn1 <- knn.cv(t(exprs(golubTrainSub)),gdf2$ALL, k = 2, prob = TRUE)
```

```
> print(table(knn1, Golub_Train$ALL))
```

Knn1	ALL	AML
ALL	26	3
AML	1	8

```
sum(knn1 == golubTrainSub$ALL)/length(golubTrainSub$ALL)
```

```
[1] 0.89
```

### **## Training KNN classifier with breast cancer dataset**

```
> knn1 <- knn.cv(X.test, y.test, k = 1, prob=TRUE)
```

```
> print(table(knn1,y.test))
```

Knn1	DM	NODM
DM	1	5
NODM	9	8

```
> knn1 <- knn.cv(X.test, y.test, k = 2, prob=TRUE)
```

```
> print(table(knn1,y.test))
```

```
y.test
```



Knn1	DM	NODM
DM	2	5
NODM	8	8

```
> knn1 <- knn.cv(X.test, y.test, k = 3, prob=TRUE)
```

```
> print(table(knn1,y.test))
```

Knn1	DM	NODM
DM	2	4
NODM	8	9

```
> ( tbl = table(predict=knn1, truth=df.test$y) )
```

Knn1	DM	NODM
DM	2	4
NODM	8	9

```
> confusionMatrix(tbl)
```

### *Confusion Matrix and Statistics*

Knn1	DM	NODM
DM	2	4
NODM	8	9

*Accuracy : 0.4783*

*95% CI : (0.2682, 0.6941)*

*No Information Rate : 0.5652*

*P-Value [Acc> NIR] : 0.8534*

*Kappa : -0.1129*

*Mcnemar's Test P-Value : 0.3865*

*Sensitivity : 0.20000*

*Specificity : 0.69231*

*PosPred Value : 0.33333*

*NegPred Value : 0.52941*

*Prevalence : 0.43478*

*Detection Rate : 0.08696*

*Detection Prevalence : 0.26087*

*Balanced Accuracy : 0.44615*

*'Positive' Class : DM*

**## Training KNN classifier with prostate cancer dataset**

```
> knn1 <- knn.cv(X.test, y.test, k=1, prob=TRUE)
```

```
> print(table(knn1,y.test))
```

y.test

Knn1	cancer	healthy
cancer	12	5
healthy	6	7

```
> knn1 <- knn.cv(X.test, y.test, k=2, prob=TRUE)
```

```
> print(table(knn1,y.test))
```

Knn1	cancer	healthy
cancer	12	7
healthy	6	5

```
> knn1 <- knn.cv(X.test, y.test, k=3, prob=TRUE)
```

```
> print(table(knn1,y.test))
```

Knn1	cancer	healthy
cancer	15	8
healthy	3	4

```
> ( tbl = table(knn1, truth=df.test$y) )
```

Knn1	cancer	healthy
cancer	12	5
healthy	6	7

```
> confusionMatrix(tbl)
```

### *Confusion Matrix and Statistics*

Knn1	cancer	healthy
cancer	12	5
healthy	6	7

*Accuracy : 0.6333*

*95% CI : (0.4386, 0.8007)*

*No Information Rate : 0.6*

*P-Value [Acc> NIR] : 0.4311*

*Kappa : 0.2466*

*Mcnemar's Test P-Value : 1.0000*

*Sensitivity : 0.6667*

*Specificity : 0.5833*

*PosPred Value : 0.7059*

*NegPred Value : 0.5385*

*Prevalence : 0.6000*

*Detection Rate : 0.4000*

*Detection Prevalence : 0.5667*

*Balanced Accuracy : 0.6250*

*'Positive' Class : cancer*